Purdue University
Purdue e-Pubs

Open Access Dissertations | Theses and Dissertations

Fall 2013

Scalable Autonomous Operations of Unmanned Assets

Sunghun Jung
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_dissertations

 Part of the Aerospace Engineering Commons, Electrical and Computer Engineering Commons, and the Mechanical Engineering Commons

Recommended Citation
Jung, Sunghun, "Scalable Autonomous Operations of Unmanned Assets" (2013). *Open Access Dissertations*. 167.
https://docs.lib.purdue.edu/open_access_dissertations/167

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Sunghun Jung

Entitled Scalable Autonomous Operations of Unmanned Assets

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Kartik B. Ariyur
Chair

Bin Yao

Inseok Hwang

Dengfeng Sun

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Kartik B. Ariyur

Approved by: David C. Anderson 07/23/2013
Head of the Graduate Program Date

SCALABLE AUTONOMOUS OPERATIONS OF UNMANNED ASSETS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Sunghun Jung

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2013

Purdue University

West Lafayette, Indiana

Once a Marine, Forever a Marine.

Republic of Korea Marine Corps

ACKNOWLEDGMENTS

Grateful appreciation is offered to professor Kartik B. Ariyur in the School of Mechanical Engineering in Purdue University for sincere discussions all the time. I also appreciate my committee, Professor Bin Yao, Professor Inseok Hwang, and Professor Dengfeng Sun. In addition to the faculty members, I also received helpful comments from labmates. Last but not least, I am indebted to my parents, Yeuntak Jung and Soonim Song, for their endless support.

PREFACE

As sensors are getting smaller and cheaper, many people are able to purchase RC planes, develop their own Unmanned Aerial Vehicle (UAV), and exchange information through online like DIYDRONE website. UAV technologies are not only for researchers in the National Aerospace labs or the National Defense and Science labs, but also opened to everyone who has basic knowledge of electronic parts as people use UAVs to take video of their town, play music [1], or even attach a machine gun [2]. Since many people do not have access to high quality sensors unlike researchers in national labs, the level of UAV development stays at certain upper limit.

This paper provides various methods of improving trajectory following, automatic taking-off and landing, and target detection and tracking using easily accessible cheap sensors. By spreading out the UAV technologies around the world, author hopes that the market of UAV is not only focused on aerospace industry, but also opened to amateur people. As more people use UAVs for diverse applications, the rate of UAV technology growth will be accelerated.

All works are supported with MATLAB/SIMULINK® based algorithms called MultiUAV System (MUAVS). MUAVS can produce trajectories for multiple UAVs by maneuvering them to avoid obstacles. Rather than using C/C++, author keeps using MATLAB/SIMULINK® to let people easily modify the algorithms for their own missions.

The outline of the paper is as follows. This paper is largely divided into Part 1, Part 2, Part 3, and Part 4. Part 1 includes chapter 1 which gives introduction including the history of UAV, individual or cooperative mission plans of UAV, challenges of using UAV, and future technologies of UAV. Part 2 includes chapter 2, 3, 4, and 5. Chapter 2 studies the method of improving position accuracy of UAV by integrating multiple GPS sensors on a Unmanned Ground Vehicle (UGV) and using

the method of image processing. Chapter 3 presents the method of improving orientation accuracy by integrating a magnetic compass and a solar compass. Chapter 4 deals with the health management of UAVs using a autonomous wireless charging ground station. Chapter 5 provides overall algorithmic error propagation and accomplishes robustness. Part 3 includes chapter 6, 7, and 8 which are focused on specific mission plans. Chapter 6 presents the minimum time Integrated Surveillance and Reconnaissance (ISR) mission by solving multiple Traveling Salesman Problem (mTSP). Chapter 7 investigates a method of achieving the maximum fuel efficiency of surveiling a given region using multiple UAVs. Chapter 8 studies methods about the guidance of stocks into pen with multiple UAVs using dynamic programming. Part 4 includes chapter 9. Chapter 9 summarizes this paper and gives future works including game theoretic tracking and reducing personnel per UAV.

Appendix A shows the experiment setup including both H/W and S/W specifications for the UAV flight demonstration. Appendix B describes the Genetic Algorithm (GA) based Traveling Salesman Problem (TSP) algorithm with a schematic of the algorithm. Appendix C gives step by step procedures of writing Block Separation Algorithm (BSA). Appendix D presents system properties of heterogeneous UAVs. Appendix E details MUAVS through four sections; MUAVS Communication Structure; MUAVS Input Process; MUAVS MATLAB/SIMULINK® Organization; MUAVS Collision Avoidance System (CAS). Appendix F presents SIMULINK® diagrams used in Chapter 8.

December 2013

Sunghun Jung

Purdue University, IN, USA

TABLE OF CONTENTS

Page

Page

APPENDICES

LIST OF TABLES

LIST OF FIGURES

Figure Page

SYMBOLS

a	acceleration
A	cross sectional area
c	wire bundle thickness
c_d	drag coefficient
d	distance
g	degree
G_d	distance array
G_t	time array
H	altitude
k	coupling coefficient
l	length or coil length/width
L	distance or inductance
m	total number of UAVs or GPS sensors
M	mutual inductance
n	number of vertices
N	number of turns in coil
n_f	total fuel consumption
o	time increase due to unexpected obstacles
Q	quality factor
r	radius of the coil
R	rotation matrix or resistance of the inductor
$\overrightarrow{r_0}$	sun vector
r_{min}	minimum turn radius
T	Direction Cosine Matrix (DCM) or sampling time or threshold value

v	velocity or noise
V	velocity
w	noise or weight or frequency or time increase due to wind
Z	impedance
ψ	yaw
θ	pitch or vehicle orientation
ϕ	roll or latitude
λ	longitude
μ_0	magnetic constant
ρ	mass density

ABBREVIATIONS

APM	ArduPilot Mega
BSA	Block Separation Algorithm
CA	Changing Altitude
CAS	Collision Avoidance System
CEP	Circular Error Probable
CL	Centroid Localization
CV	Changing Velocity
DCM	Direction Cosine Matrix
ECEF	Earth-Centered Earth-Fixed
EKF	Extended Kalman Filtering
EM	Electromagnetic
FCC	Federal Communication Commission
FSR	Force Sensitive Resistor
GA	Genetic Algorithm
GCS	Ground Control Station
HSS	Hemispherical Solar Sensor
INS	Inertial Navigation System
ISR	Integrated Surveillance and Reconnaissance
KF	Kalman Filtering
KVRD	K-means Voronoi Region Division
LSE	Least Square Estimation
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
mTSP	Multiple Traveling Salesman Problem
MUAVS	MultiUAV System
NP-hard	Non-deterministic Polynomial-time hard

NEA	Node Exchange Algorithm
PSA	Path Sliding Algorithm
PEA	Polygon Extension Algorithm
PIR	Passive Infrared motion sensors
RF	Radio Frequency
RGA	Region Growing Algorithm
RMSE	Root Mean Square Error
ROA	Remotely Operated Aircraft
ROS	Region of Support
RPA	Remotely Piloted Vehicle
RSTA	Reconnaissance, Surveillance, and Target Acquisition
TPT	Three Pixel Theorem
TRN	Terrain Referenced Navigation
TSP	Traveling Salesman Problem
UA	Unmanned Aircraft
UAAV	Unmanned Autonomous Aerial Vehicle
UALV	Unmanned Aerial Logistics Vehicle
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UCAV	Unmanned Combat Air Vehicle
UGV	Unmanned Ground Vehicle
URD	Uniform Region Division
WCL	Weighted Centroid Localization

ABSTRACT

Jung, Sunghun. Ph.D., Purdue University, December 2013. Scalable Autonomous Operations of Unmanned Assets. Major Professor: Kartik B. Ariyur, School of Mechanical Engineering.

Although there have been great theoretical advances in the region of Unmanned Aerial Vehicle (UAV) autonomy, applications of those theories into real world are still hesitated due to unexpected disturbances. Most of UAVs which are currently used are mainly, strictly speaking, Remotely Piloted Vehicles (RPA) since most works related with the flight control, sensor data analysis, and decision makings are done by human operators. To increase the degree of autonomy, many researches are focused on developing Unmanned Autonomous Aerial Vehicle (UAAV) which can takeoff, fly to the interested area by avoiding unexpected obstacles, perform various missions with decision makings, come back to the base station, and land on by itself without any human operators.

To improve the performance of UAVs, the accuracies of position and orientation sensors are enhanced by integrating a Unmanned Ground Vehicle (UGV) or a solar compass to a UAV; Position sensor accuracy of a GPS sensor on a UAV is improved by referencing the position of a UGV which is calculated by using three GPS sensors and Weighted Centroid Localization (WCL) method; Orientation sensor accuracy is improved as well by using Three Pixel Theorem (TPT) and integrating a solar compass which composed of nine light sensors to a magnetic compass. Also, improved health management of a UAV is fulfilled by developing a wireless autonomous charging station which uses four pairs of transmitter and receiver magnetic loops with four robotic arms. For the software aspect, I also analyze the error propagation of the proposed mission planning hierarchy to achieve the safest size of the buffer zone.

In addition, among seven future research areas regarding UAV, this paper mainly focuses on developing algorithms of path planning, trajectory generation, and cooperative tactics for the operations of multiple UAVs using GA based multiple Traveling Salesman Problem (mTSP) which is solved by dividing into m number of Traveling Salesman Problems (TSP) using two region division methods such as Uniform Region Division (URD) and K-means Voronoi Region Division (KVRD). The topic of the maximum fuel efficiency is also dealt to ensure the minimum amount fuel consumption to perform surveillance on a given region using multiple UAVs. Last but not least, I present an application example of cattle roundup with two UAVs and two animals using the feedback linearization controller.

PART 1: INTRODUCTION

CHAPTER 1: INTRODUCTION

1.1 Definition of Unmanned Aerial Vehicle (UAV)

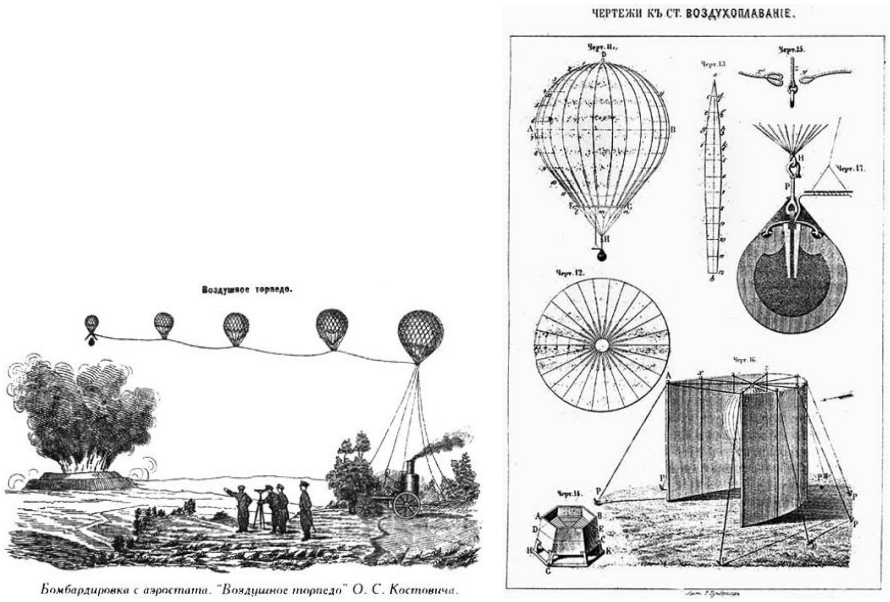
UAV is defined as powered, aerial vehicle that does not carry a human operator, uses aerodynamic forces to provide vehicle lift, can fly autonomously or be piloted remotely, can be expendable or recoverable, and can carry a lethal or nonlethal payload [3] and the fact that UAVs can be recovered after the mission completion differentiates UAVs from missiles. Many names have been appeared such as Unmanned Aircraft (UA), Drone, Remotely Piloted Vehicle (RPA), Remotely Operated Aircraft (ROA), and Unmanned Combat Air Vehicle (UCAV), but UAV is the most commonly used name. Some researchers misuse Unmanned Aerial System (UAS) as UAV, but UAS is more upper level category which contains Unmanned Aircraft (UA), Ground Control Station (GCS), control link between UA and GCS, and other additional equipments. Technologies of UAV is getting concentrated on the aspect of autonomy [4], so Unmanned Autonomous Aerial Vehicle (UAAV) will be perhaps more appropriate name for the future UAV which will be fully autonomous.

1.2 Development Chronicle of UAV

From the late nineteenth century to current, UAV development history can be divided into largely four eras by relating to war history as below.

1. The 1^{st} era (1849 - 1937): Austria and Italy war, World War I
2. The 2^{nd} era (1937 - 1945): World War II
3. The 3^{rd} era (1945 - 1991): Cold war, Vietnam war
4. The 4^{th} era (1991 - current): Afghanistan war, Iraq war

The 1^{st} era was the period between 1849 and 1938 including Austria and Italy war and World War I. The world first pilotless aircraft was developed when Austria military used series of unmanned balloons (Figure 1.1) loaded with explosives to attack Venice, Italy in 1849 [5].



(a) Series of balloons loaded with a bomb. (b) Schematics of the system.

Figure 1.1.Austrians balloon attack during the 1^{st} era.

In 1900, Nikola Telsa, the founder of the wireless remote control technology [6], introduced a wireless controlled airship (Figure 1.2). Since then, many additional designs were attempted such as Aerial Target in 1916 [7], Aerial Torpedo (also called as Flying Bomb) in 1917 [8], and Kettering Bug in 1918 [9].

Following the 1^{st} era, the 2^{nd} era started during World War II from 1939 to 1945. During this period, Reginald Denny open business of RC airplane and started to sell mass-produced RC planes. At the same time, the US Navy began to produce N2C-2 (Figure 1.3(a)) which was controlled by TG-2 (Figure 1.3(b)) aircraft in 1937 [10]. As World War II was prolonged to late 1940's, assault drones such as Project Fox started to be used heavily [10]. Project Fox was the first UAV which carried a camera to enhance the controllability.

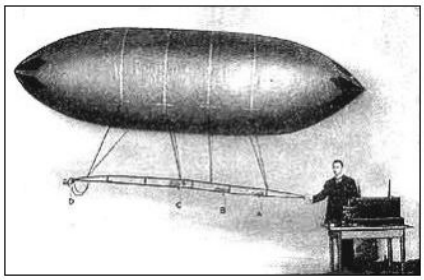


Figure 1.2.Nikola Telsas wireless controllable airship during the 1^{st} era.

(a) N2C-2.

(b) TG-2.

Figure 1.3.N2C-2 and TG-2 by the US Navy during the 2^{nd} era.

The 3^{rd} era was a period between 1946 and 1991 including Cold war and Vietnam war. UAVs became more sophisticated than ever and functions of UAVs were diversified from target drone to nuclear tests and reconnaissance. The most representative target drones were OQ-2, OQ-19 (Figure 1.4(a,b))/KD2R Quail, and MQM-33 (Figure 1.4(c))/MQM-36 Shelduck.

(a) OQ-19A.

(b) OQ-19B.

(c) MQM-33C.

Figure 1.4.The most representative target drones during the 3^{rd} era.

Modern society is in the 4^{th} era which started in 1992 and includes Afghanistan war and Iraq war. Many researches are mostly focusing on making fully autonomous UAV throughout the world and this flow resulted in thousands of UAV designs. The most famous UCAV in the world is the Predator RQ-1L (Figure 1.5(a)) UAV which was produced by General Atomics and heavily deployed during Afghanistan war and Iraq war. Researchers attempted to solve the fuel limitation problem by designing solar powered UAV (Figure 1.5(b)) and beam powered UAV (Figure 1.5(c)), or to design an autonomous wireless charging ground station to increase the degree of autonomy.

(a) The Predator RQ-1L. (b) Helios solar powered UAV. (c) The concept of beam powered UAV.

Figure 1.5.Various types of UAVs during the 4^{th} era.

1.3 Functions of UAV

With widespread of UAVs into the human life, functions of UAVs become much broader than before in modern society and the general functions can be approximately divided into six categories such as target and decoy, reconnaissance, combat, logistics, research and development, and civil and commercial UAVs.

1. Target and decoy: UAVs are used as a target for both antiaircraft gunners and pilots to train them [11].

2. Reconnaissance: UAVs are used to alleviate pilots from surveilling regions by searching targets, recording ground videos, and saving battlefield information [12].

3. Combat: Beyond the reconnaissance function, UAVs are armed weapons to perform attacks in highly risky areas and called UCAV [13].

4. Logistics: When ground vehicles are not applicable, UAVs are used to suppy foods, munitions, etc by landing on or using parachute systems [14]. These UAVs are specifically called Unmanned Aerial Logistics Vehicles (UALV).

5. Civil and Commercial UAVs [15]: UAVs are also used by civilians to perform commercial aerial surveillance (ex. oil [16], gas [17], and mineral exploration and production [18]), transport [19], scientific research [20], search and rescue [21], and conservation of wildlife [22].

1.4 Individual or Cooperative Missions of UAV

In the contrast to the classical individual mission planning using one UAV, cooperative mission planning using not only several UAVs but also several UGVs (Unmanned Ground Vehicle) comes into the spotlight due to synergy effects. As performance boundaries of UAVs are getting bigger with the help of up to date technologies on sensors and autonomy, missions become much more complicated which result in the necessity of the UAVs and UGVs cooperation. Multiple simultaneous UAV operations require smarter way of maneuvering UAVs, resources, and surrounding information which result in attentions on global information, resource management, and robustness [23].

1.5 Challenges of UAV

The scope and complexity of UAV missions have made autonomous real-time operation in unstructured environments elusive [24, 25]. Hence, the dependence on multiple human operators per UAV is perhaps the largest cost component of operating them [26]. Operator attention and effort are subject to fatigue and error [27], operators are subject to severe stresses, especially in combat operations, something that can

be avoided if tasks are suitably automated [28]. There are many efforts under way to increase the autonomy of UAVs [23,29,30]. The difficulties arise from the propagation of several uncertainties, many dynamic, into the performance of these systems–map errors, modeling errors, sensor errors, errors of actuation, wind gusts, electromagnetic (EM) and acoustic interference, enemy maneuvers, and cyberattacks [31]. Arbitrary combinations of these uncertainties make it impossible to predict the performance of most battlefield systems as they are operated today.

1.6 Future Technology of UAV

The world famous UCAV, MQ-1 Predator (Figure 1.6(a)), is controlled by MD-1 ground equipment (Figure 1.6(b)). One of the MD-1 series, MD-1D, can control up to 4 UAVs simultaneously but it requires total 55 operators including 1 pilot, 4 sensor operators, and the rest operators for a Predator Primary Satellite Link and etc which is very heavy duty work. Due to heavy requires of manpower on UAV operations, recent researches are flowing to the autonomy area of UAV. To develop fully autonomous UAV, UAAV, mainly seven research areas need to be accomplished; data fusion [32], communications [33], path planning [34–36], trajectory generation (or, motion planning) [37,38], trajectory regulation [39], task allocation and scheduling [40,41], and cooperative tactics [23].

(a) MQ-1 predator. (b) MD-1 ground equipment.

Figure 1.6.MQ-1 predator and MD-1 control unit.

1.7 Concluding Remarks

This paper is focused on improving autonomous functions of path planning, trajectory generation, and cooperative tactics among seven research areas dealt in the above. Also, to help the completion of the autonomous functions, position and orientation sensor improvements and autonomous wireless charging system are handled. Developing an fully autonomous aerial vehicle involves state of the art technologies of the various fields from Mechanical engineering and Electorial engineering to Psychology, so there are huge number of opened up questions for me to solve in the future.

PART 2: UNCERTAINTIES, MITIGATION, AND ROBUSTNESS

CHAPTER 2: POSITION SENSING

I improve the performance of low-quality GPS on UAVs through use of multiple GPS modules on a UGV within line of sight of the UAV. Moreover, in my analysis and experiments, the GPS modules on the UGV are also low cost and low weight. The UGV sends a GPS correction to the UAV on the basis of the distance from the UAV to the UGV as measured by scaling of a standard image pattern stuck on its back. Geolocation of both UGV and UAV are performed through use of extended Kalman filters integrating GPS aided INS. The positioning error is reduced by a factor of 2.3 in simulation studies and a factor of 1.6 in experiment when 3 GPS sensors are used on the UGV. This is better than what one can get through pure averaging of the GPS sensors in the presence of noise in measuring the UAV-UGV distance. I show how my exploitation of geometry improves performance as more GPS sensors are used.

2.1 Background and Motivation

Close swarming operations of Unmanned Aerial Vehicles (UAV) need accurate geolocation of each UAV for collision avoidance. Many of the tight swarming demonstrations thus far have been indoors [42, 43]. Only a few works demonstrate outdoor performance, where however inter-vehicle separation is relatively large [44, 45]. GPS aided INS is the standard method of geolocation for aerial vehicles [46–49]. GPS aiding eliminates the large drift inherent in dead-reckoning ($\propto t^3$) [50]. Vision-based terrain referenced navigation (TRN) method is another method of reducing the drift of dead reckoning [51, 52].

Schrader [53] improved geolocation through averaging data from multiple GPS receivers fixed to the ground. The data is collected asynchronously and averaged, something unsuitable for real-time use on a moving platform since GPS signals are

subject to time varying disturbances due to satellite motion, weather changes, and magnetic field changes [54]. Another problem was that all four GPS sensors are permanently placed at one location, though most of GPS applications in real life are to track moving objects which limit the amount of GPS data collection at one location.

Weighted Centroid Localization [55] uses multiple RF range measurements but does not consider the geometry of the sensor arrangement or any motion of the sensor platform. To ameliorate these deficiencies, I integrate the GPS sensors on the UGV exploiting the geometry of their placing and the statistics of individual sensors along with UGV motion. Once GPS data are improved on a UGV, a UAV which carries one low-quality GPS sensor is maneuvered to reference the improved UGV GPS data by detecting a mark on the UGV to enhance UAV's own GPS accuracy. Finally, the extended Kalman filter (EKF) is applied to achieve nonlinear state estimation of GPS data by integrating the INS and GPS data [56]. This method is unique compare to the other existing GPS aiding techniques, e.g. Differential GPS (DGPS), Local Area Augmentation System (LAAS), Wide Area Augmentation System (WAAS), etc, in the sense that my method uses moving ground reference stations in contrast to the listed other methods which use augmented network systems based on static ground stations.

A variety of applications and individuals stand to benefit from my approach; individual hobbyists, law enforcement, and movie makers can obtain significantly improved geolocation at low cost as individual GPS receivers on the UGV cost only a few cents. It will enable the use of laser and RF powering of small UAVs through improving geolocation of the UAV in line of sight of the powering source. This also makes this kind of powering less risky as a power carrying beam can destroy sensors or structural elements in a small UAV [57–59]. Finally, tighter swarms of vehicles and resulting applications will be enabled at low cost.

I introduce a method to improve the UGV GPS accuracy in Section 2.2, a method to improve the UAV GPS accuracy in Section 2.3, altitude and attitude limit to

detect a mark by the UAV in Section 2.4, the UGV and UAV controllers in Section 2.5, simulation results of the GPS improvement of the UGV and UAV in Section 2.6, experimental results in Section 2.7, and concluding remarks in Section 2.8.

2.2 UGV GPS Improvement

In Figure 2.1, $P_{1,k}$, $P_{2,k}$, and $P_{3,k}$ represent the data of GPS_1, GPS_2, and GPS_3 sensors represented as red dots, L is the length of the bar attached to the center of UGV ($2m$), and R_{GPS} represents the radius of resolution boundary of a MediaTek MT3329 GPS module (horizontal position accuracy as $3.0m$ CEP).

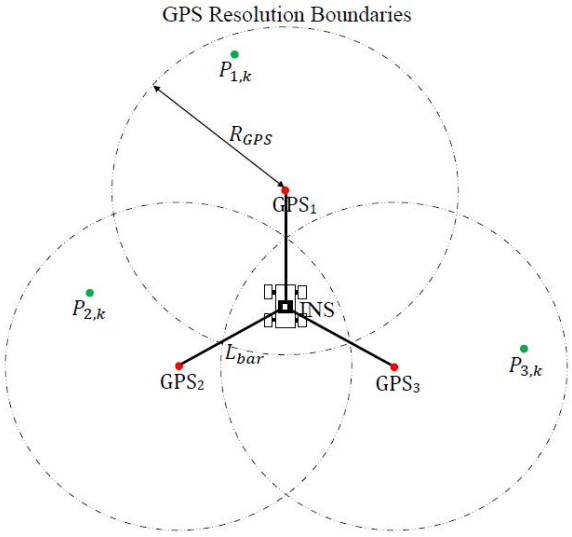


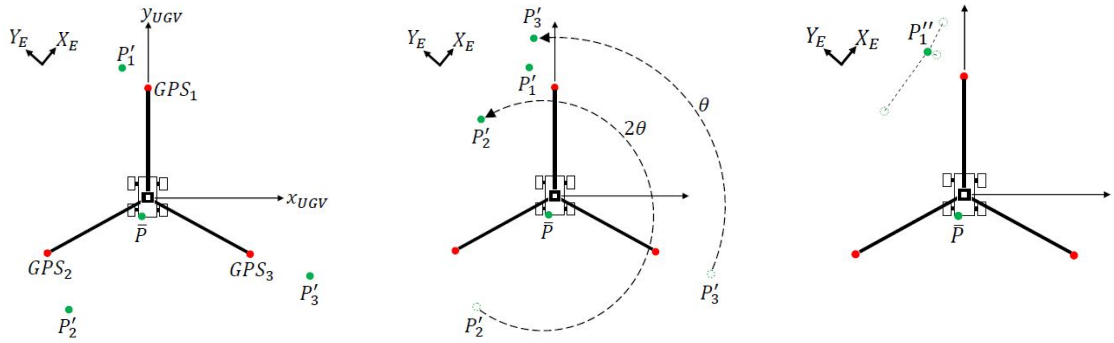
Figure 2.1.Three GPS modules attached on a UGV.

Most of all, coordinate transformation should be done for the latitude, longitude, and altitude of GPS data as $(x, y, z) = T_1\left[(lat, lon, alt)\right]$ where $T_1 \equiv T_{(lat,lon,alt)\rightarrow(x,y,z)}$. Then, since I am using multiple GPS sensors which are located at a specific location, I can improve one GPS sensor location with the rest of GPS sensors as,

$$
\begin{aligned}
P_i'' &= \overline{P} + \frac{1}{m}\left(P_i' + \sum_{j=1, i\neq j}^{m} R_z(\theta) P_j'\right), \qquad (2.1)\\
P_i' &= \frac{\sum_{k=1}^{n} w_{i,k} P_{i,k}}{\sum_{k=1}^{n} w_{i,k}} - \overline{P},\\
P_j' &= \frac{\sum_{k=1}^{n} w_{j,k} P_{j,k}}{\sum_{k=1}^{n} w_{j,k}} - \overline{P},\\
\overline{P} &= \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n} P_{i,k},
\end{aligned}
$$

where P_i is the ith GPS data, m is the total number of GPS sensors, n is the total number of the GPS data collected at one position, $R_z(\theta)$ is the rotation matrix where θ can be calculated as $\theta = \frac{2\pi}{m}$, $w_{i,k}$ is the weight of the ith GPS sensor at data sample k where $w_{i,k}$ can be calculated as $w_{i,k} = \frac{1}{(d_{i,k})^g}$, $d_{i,k}$ is the distance between a mean of enough number of the ith GPS sensor data and the kth data of the ith GPS sensor, and g is a degree. When the degree g is set to be zero, the WCL becomes the Centroid Localization (CL) [55]. In this chapter, the degree g is set to be 1 but I need to use the degree factor as zero when high number of GPS data are involved in WCL. The reason is that when high number of GPS data are involved and applied to WCL, it is natural that the centroid of all GPS data is getting close to the exact centroid of three GPS sensors and WCL is getting closer to CL. So, I should use the degree as zero when I use high number of GPS data and use appropriate degree value which brings low mean position error when I use low GPS data. Also, I approximate n as a static constant proportional to the frequency of the GPS sensor divided by the velocity of the UGV in meters/second, $n = \frac{f_s}{v}$. It works so long as motion of the UGV is slow enough so that UGV displacement is less than GPS sensor error. Estimated locations of the other GPS sensors except the reference GPS sensor are moved using

the rotation matrix about $\overline{P}$ to near the reference GPS sensor, and then I average those to calculate the improved location of the reference GPS sensor (Figure 2.2).



(a) Step 1. (b) Step 2. (c) Step 3.

Figure 2.2.Method of improving GPS data accuracy (top view).

Most of the time, the UGV moves on an uneven surface (Figure 2.3), so I need to rotate the GPS data about $\overline{P}$ to be horizontal to the ground using roll (ϕ), pitch (θ), and yaw (ψ) measured from an IMU sensor mounted on the UGV center as,

$$P_i''' = R_{xyz}\left(P_i'' - \overline{P}\right) + \overline{P}, \tag{2.2}$$

where R_{xyz} is a rotation matrix $= R_z(\psi)R_y(\theta)R_x(\phi)$. Lastly, past q number of GPS data are used to improve the current GPS data as,

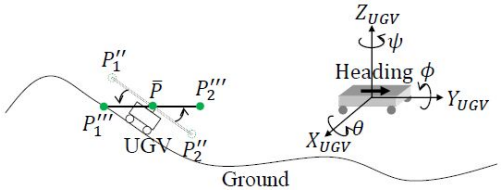


Figure 2.3.Rotation of the inclined GPS data to be horizontal to the ground (side view).

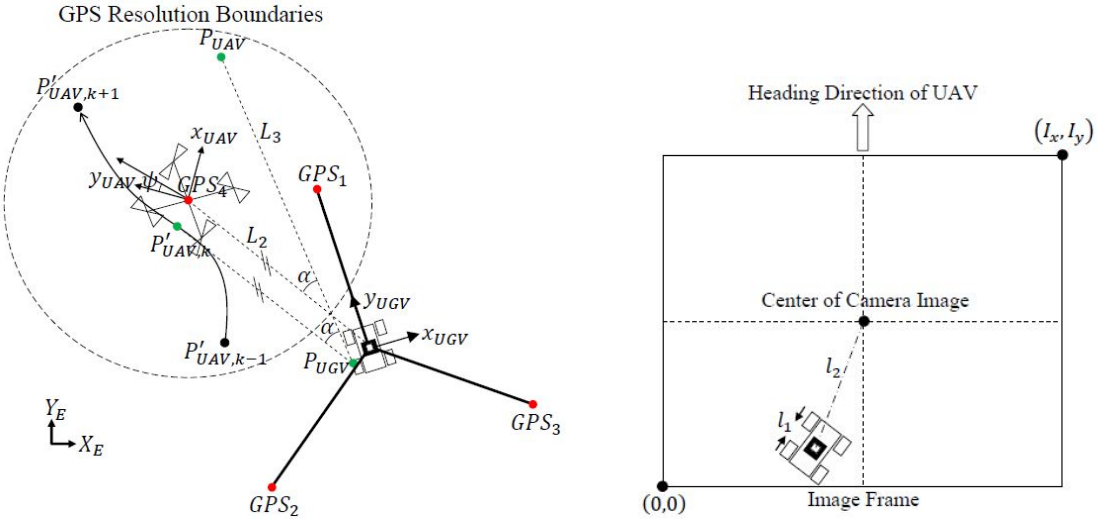
$$P_{UGV,k} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{q}\sum_{k=n-(q-1)}^{n} P_{i,k}'''\right), \tag{2.3}$$

where q is the number of past GPS data, n is an integer ($\geq q$), and the P_{UGV} is the reference point which the UAV uses to improve own GPS accuracy. If all previous equations are combined, P_{UGV} becomes

$$P_{UGV,k} = \frac{1}{mq}\sum_{i=1}^{m}\sum_{k=n-q+1}^{n}\left(R_{xyz}\left(\frac{1}{m}P'_i + \frac{1}{m}\sum_{j=1,i\neq j}^{m} R_z(\theta)P'_j\right) + \overline{P}\right)_k . \tag{2.4}$$

2.3 UAV GPS Improvement

Figure 2.4(a) describes a method of calculating the estimated position of the UAV at time instant k, that is, $P'_{UAV,k}$ (Equation 2.5). With help of the NyARToolkit algorithm [60], a black square mark on the UGV can be detected and positions of four corners of the mark can be obtained. I set up an equation to calculate the estimated UAV location, $P'_{UAV,k}$, in Earth-centered/Earth-fixed (ECEF) coordinates as



(a) Communication between the UAV and UGV (top view).

(b) Calculation of lengths l_1 and l_2

Figure 2.4.The l_1 and l_2 to calculate L_2 in communication diagram between the UAV and UGV.

$$P'_{UAV,k} = \left[\frac{L_2}{L_3}R_z(\alpha)\left(P_{UAV} - P_{UGV}\right) + P_{UGV} + \cos(|\psi|)V_{UAV}\Delta t\right]_k, \quad (2.5)$$

where P'_{UAV} is the finally achieved position of the UAV in ECEF coordinates, $R_z(\alpha)$ is the rotation matrix which rotates a vector $\overrightarrow{P_{UGV}, P_{UAV}}$ in the amount of α, P_{UAV} is the improved GPS data of the UAV using Equation 2.3, L_2 is a distance from the UAV_{center} to the UGV_{center}, L_3 is a distance from the P_{UAV} to the P_{UGV}, V_{UAV} can be calculated as $V_{UAV} = T_2^T[v_{px}, v_{py}]^T$, Δt is a sampling time, T_2 is a Direction Cosine Matrix (DCM), and v_{px}, v_{py} are measured airspeeds in x and y direction of the UAV. In Equation 2.5, the sampling time ΔT is set to $0.1s$ since the MediaTek MT3329 GPS sensor has the maximum 10Hz update rate. Also, only XY plane in ECEF Cartesian coordinates is considered since the altitude difference between the UAV and UGV is set to be $3m$ and the altitude change compared to XY direction is relatively ignorable. This explains why V_{pz} is set to zero. Also, $\cos(|\psi|)V_{UAV}\Delta t$ appears since the UAV flies further during the time while the GPS signal is transmitted to the microcontroller [61].

To solve Equation 2.5, the L_2 and α should be known. The distance, L_2, can be calculated by using l_1 and l_2 (Figure 2.4(b)) as $L_2 = \frac{l_2 L_1}{l_1}$, where l_1 is the image length of the square mark edge, l_2 is the image length from the image center to the UGV center, and L_1 is the metric size of the square mark edge (Figure A.1(c)). Since the units of L_2 and L_3 are different, these are used only for calculating the angle α. Once P'_{UAV} is found (Equation 2.5), I convert the ECEF Cartesian coordinates back to the latitude (ϕ), longitude (λ), and altitude (H). If multiple UAVs are in operation, improved positions of other UAVs can also be calculated by individually applying the above procedures.

2.4 Altitude and Attitude Limit for Mark Detection

To simulate the detection of a mark on the UGV by a camera attached under the UAV, it is assumed that a mark is detected if one of the mark corners (Figure 2.5)

among m_1'', m_2'', m_3'', and m_4'' is inside the camera image drawn with four corners (I_1, I_2, I_3, and I_4). Here, the shape and size of the camera image changes as the attitude and altitude of the UAV changes. I_0 is fixed to the ground level.

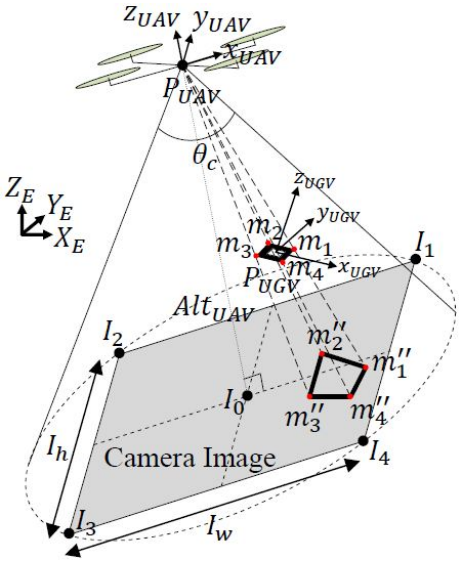


Figure 2.5.A mark on the UGV is detected by a camera under the UAV.

The size of the mark image should be in certain boundaries $\{l_{min}, l_{max}\}$ to be detected by the UAV since too small or too big mark cannot be recognized by the NyARToolkit algorithm and the mark image size can be adjusted by changing the altitude of the UAV. To find the maximum and minimum allowable altitude of the UAV to detect the mark on the UGV, I first need to define the location of the mark as

$$m_i = \begin{bmatrix} m_{i,x} \\ m_{i,y} \\ m_{i,z} \end{bmatrix} = \begin{bmatrix} P_{UGV,x} + \frac{1}{\sqrt{2}} L_1 \cos\left(\theta_h - \frac{\pi}{4} + \frac{\pi}{2}(i-1)\right) \\ P_{UGV,y} + \frac{1}{\sqrt{2}} L_1 \sin\left(\theta_h - \frac{\pi}{4} + \frac{\pi}{2}(i-1)\right) \\ P_{UGV,z} \end{bmatrix}_i , \tag{2.6}$$

for $i \in \{1, 2, 3, 4\}$, L_1 is the metric size of the square mark edge, and θ_h is the heading angle of the UGV. Then, I define vectors from the center to each corner of

the mark as $v_i = m_i - P_{UGV}$. Using the rotation matrix, I can represent the locations of the mark corners as

$$\begin{aligned} m_i' &= R(\theta_i)v_i + P_{UGV}, \\ \theta_i &= \begin{bmatrix} \phi_{UGV} - \phi_{UAV} \\ \theta_{UGV} - \theta_{UAV} \\ \psi_{UGV} - \psi_{UAV} \end{bmatrix}_i , \end{aligned} \tag{2.7}$$

where ϕ is roll, θ is pitch, and ψ is yaw. I take only x and y components of m_i' and let z component to be same with $P_{UGV,z}$. Now, using the ratio between the mark size in real and the mark size in camera image, I calculate the location of the mark in camera image as,

$$m_i'' = m_i' \frac{\frac{1}{2}\sqrt{I_w^2 + I_h^2}}{Alt_{UAV} \tan\left(\frac{\theta_c}{2}\right)}, \tag{2.8}$$

where I_w is the width of the image frame, I_h is the height of the image frame, Alt_{UAV} is the altitude of the UAV, and θ_c is the camera view angle attached under the UAV. With the mark corners in image, m_i'', I can set the boundaries for the mark as

$$l_{min} \leq \parallel m_i'' m_j'' \parallel \leq l_{max}, \tag{2.9}$$

where $i \neq j$ and l_{min}, l_{max} are the minimum and maximum image lengths to be detected. By substituting Equation 2.8 into Equation 2.9, I get

$$l_{min} \leq \frac{\frac{1}{2}\sqrt{I_w^2 + I_h^2}}{Alt_{UAV} \tan\left(\frac{\theta_c}{2}\right)} \parallel m_i' m_j' \parallel \leq l_{max}. \tag{2.10}$$

By rearranging Equation 2.8, I finally get

$$\frac{\frac{1}{2}\sqrt{I_w^2 + I_h^2}}{l_{max} \tan\left(\frac{\theta_c}{2}\right)} \parallel m_i' m_j' \parallel \leq Alt_{UAV} \leq \frac{\frac{1}{2}\sqrt{I_w^2 + I_h^2}}{l_{min} \tan\left(\frac{\theta_c}{2}\right)} \parallel m_i' m_j' \parallel . \tag{2.11}$$

Not only the altitude, but also the attitude of the UAV limits the detection of the mark since too much tilted mark cannot be recognized by the detection algorithm. So, I need secondary boundaries as

$$\begin{aligned} \mid \phi_{UGV} - \phi_{UAV} \mid &< \xi_{max}, \\ \mid \theta_{UGV} - \theta_{UAV} \mid &< \xi_{max}, \end{aligned} \tag{2.12}$$

where ξ_{max} is the maximum angle which can be used for the image detection algorithm to detect the mark and it is about 75° in my case. Both properties, Equation 2.11 and Equation 2.12, should be satisfied to detect the mark.

Mark detection error rate is dependent on the distance between the camera and the mark and the image frame rate which can be expressed as $\dot{E}_m = \frac{l}{f}$ where l is the distance and f is the frame rate.

2.5 Dynamics and Control of UGV and UAV

Simulating a UAV tracking a UGV which is moving from a start point to a goal point can be done by integrating a UGV controller and a UAV controller (Figure 2.6). A simple PD controller is used for the UGV and the UAV. Since the PD controller is well known controller, I skip explanation of the PD controller.

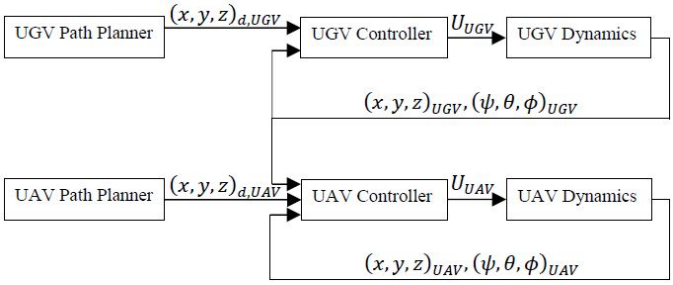


Figure 2.6.UAV and UGV control schematic.

2.5.1 UGV Dynamics

The UGV controller takes over the generated path, $(x, y, z)_d$, from the path planner [62] and results control inputs, U_{UGV} (Figure 2.6). Then, the UGV Dynamics uses the U_{UGV} to result $(x, y, z)_{UGV}$ and $(\psi, \theta, \phi)_{UGV}$ where the attitude of UGV is calculated based on the slope of the ground shape where the UGV is located. The vehicle dynamic model can be derived as,

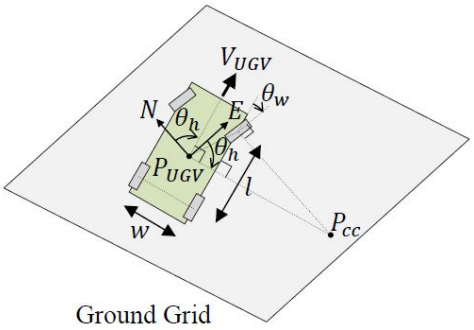


Figure 2.7.UGV System Modeling.

$$
\begin{aligned}
P'_{UGV} &= R_{xy}(\theta)(P_{UGV} - P_{cc}) + P_{cc}, \\
&= R_{xy}(\dot{\theta}_{cc}\Delta t)(P_{UGV} - P_{cc}) + P_{cc}, \\
&= R_{xy}\left(\frac{V_{UGV}\Delta t}{\parallel P_{UGV}, P_{cc} \parallel}\right)(P_{UGV} - P_{cc}) + P_{cc},
\end{aligned}
\tag{2.13}
$$

where P'_{UGV} is the next position of the UGV after a sampling time Δt, P_{UGV} is the current position of the UGV, $R_{xy}(\theta)$ is the rotation matrix on the xy plane, P_{cc} is a center of curvature of the UGV rotation, $\dot{\theta}_{cc}$ is the angular velocity about the P_{cc}, and V_{UGV} is the velocity of the UGV (Figure 2.7). The N and E represent the north and east and the z component of the UGV is set to be equal with the altitude of the ground grid. In Equation 2.13, the $\parallel P_{UGV}, P_{cc} \parallel$ and P_{cc} can be calculated as

$$\begin{aligned} \parallel P_{UGV}, P_{cc} \parallel &= \frac{1}{2}\left(w + l\tan\left(\frac{\pi}{2} - \theta_w\right)\right), \\ P_{cc} &= P_{UGV} + \parallel P_{UGV}, P_{cc} \parallel [\cos\theta_h, \sin\theta_h]^T. \end{aligned} \tag{2.14}$$

Attitude of the UGV is dependent on the ground grid where the UGV is currently located at and it can be simplified using the roll (ϕ_{UGV}), pitch (θ_{UGV}), and yaw (ψ_{UGV}) angles of the UGV (Figure 2.3) as,

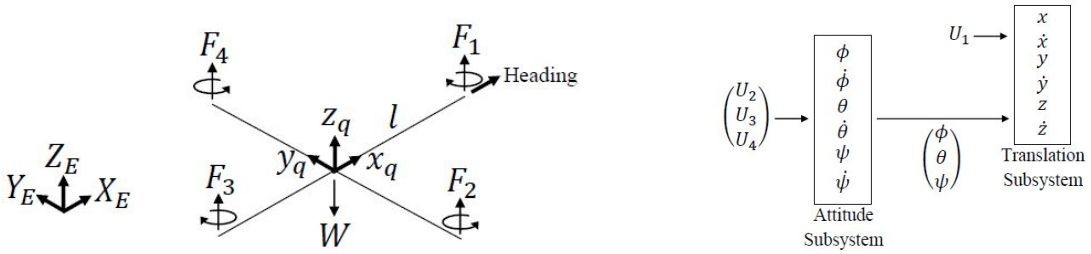
$$\begin{aligned} \phi_{UGV} &= \arctan\left(\frac{r_{21}}{r_{11}}\right), \\ \theta_{UGV} &= \arctan\left(\frac{-r_{31}}{r_{32}^2 + r_{33}^2}\right), \\ \psi_{UGV} &= \arctan\left(\frac{r_{32}}{r_{33}}\right), \end{aligned} \tag{2.15}$$

where r_{ij} is the component of the rotation matrix, R, which can be calculated using Rodrigues' formula [63].

2.5.2 UAV Dynamics

The configuration and governing control inputs of the quadrotor are described in Figure 2.8.

Quadrotor UAV dynamics are derived in [64] as



(a) Quadrotor configuration.

(b) Governing controller structure of the quadrotor.

Figure 2.8.Quadrotor configuration and governing control inputs.

$$\dot{V} = \begin{bmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{V}_3 \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (c\phi s\theta c\psi + s\phi s\psi)\frac{1}{m} \\ (c\phi s\theta s\psi - s\phi c\psi)\frac{1}{m} \\ (c\phi c\theta)\frac{1}{m} \end{bmatrix} U_1 + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \quad (2.16)$$

$$\dot{w} = \begin{bmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \end{bmatrix} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J}{I_x}\dot{\theta}\Omega + \frac{l}{I_x}U_2 \\ \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) - \frac{J}{I_y}\dot{\phi}\Omega + \frac{l}{I_y}U_3 \\ \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \frac{l}{I_z}U_4 \end{bmatrix},$$

where x, y, and z are the UAV position, ϕ, θ, and ψ are the roll, pitch, and yaw, $c\theta$ and $s\theta$ represent $\cos\theta$ and $\sin\theta$, $I_{x,y,z}$ is body inertias, J is a propeller inertia, and l is a lever (i.e., propeller length). The system's inputs (U_1, U_2, U_3, U_4 and Ω) can be rewritten as

$$\begin{aligned} U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \quad (2.17) \\ U_2 &= b(\Omega_4^2 - \Omega_2^2), \\ U_3 &= b(\Omega_3^2 - \Omega_1^2), \\ U_4 &= d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2), \\ \Omega &= \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3, \end{aligned}$$

where Ω_i is a rotor speed, b is a thrust factor, and d is a drag factor.

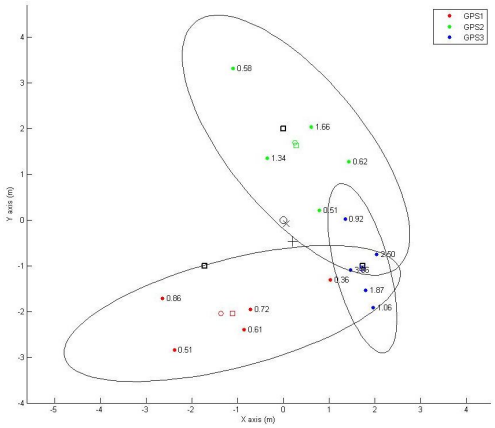
2.6 Simulation

2.6.1 UGV GPS Improvement

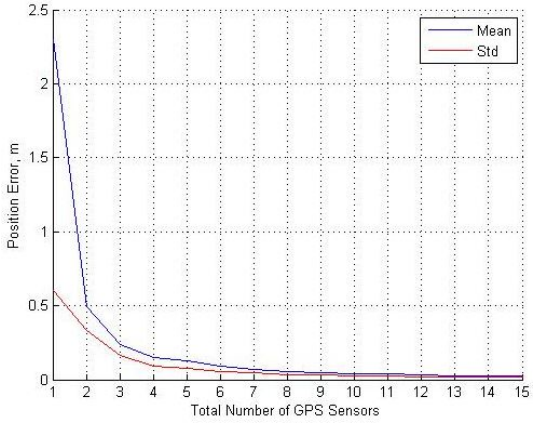
The amount of improvement using Equation 2.4 is demonstrated in here using MATLAB® by performing 100 times of simulations (Figure 2.9). Three different colored dots represent three GPS sensor data on the UGV, black elliptic circles cover 95% population of each GPS sensor's data, thick black □ represents the locations of three GPS sensors, values right next to the dots represent weights for the WCL, small □ with three different colors represent the mean values of each GPS data, small ∘ with three different colors represent the mean values calculated using the WCL, a black ◯ represents a mean value of three GPS sensor locations, a black × represents a center calculated from Equation 2.4, and a black + represents a mean of the small □ with three different colors. The distance from a black ◯ to a black + is calculated as $0.3434m$ and distance from a black ◯ to a black × as $0.0692m$. This indicates that Equation 2.4 indeed improves the accuracy about 5 times. Overall, position error decreases drastically as the number of GPS sensors increase upto three sensors (Figure 2.9(b)).

2.6.2 UAV GPS Improvement

I let the UGV drive from $(0, 0)$ to $(10, 10)$ while the UAV tracks it (Figure 2.10(a)). Here, the ground grids are randomly generated within the range of $0m$ to $1m$ altitude, the UGV is represented as a red rectangle with a sonar range finder represented as green lines, and the UAV is represented with circles and lines [65,66]. The distribution of the three GPS sensor data lies within less than $3m$ (CEP, 50%) horizontal position accuracy.

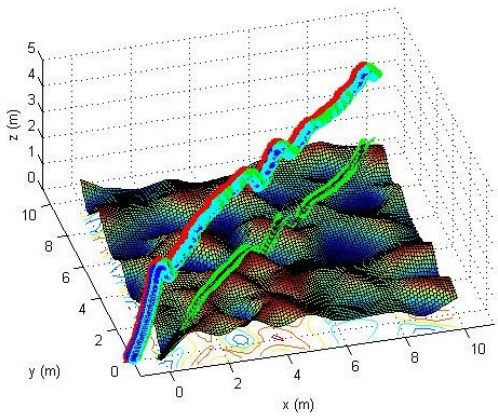


(a) WCL method.

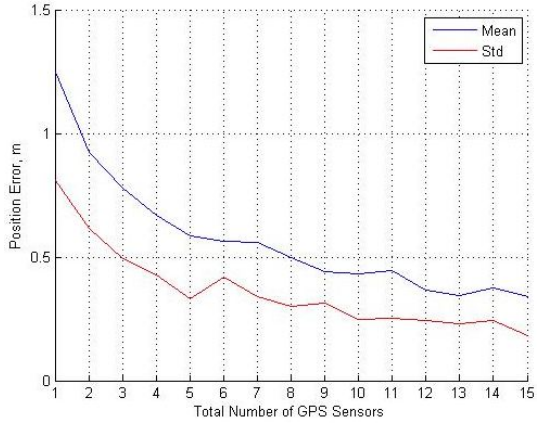


(b) Position error versus number of GPS sensors. N number of GPS sensors are assumed to be placed on the ground.

Figure 2.9.Simulation of GPS accuracy improvement.



(a) Trajectories of the UAV and UGV from $(0, 0)$ to $(10, 10)$.

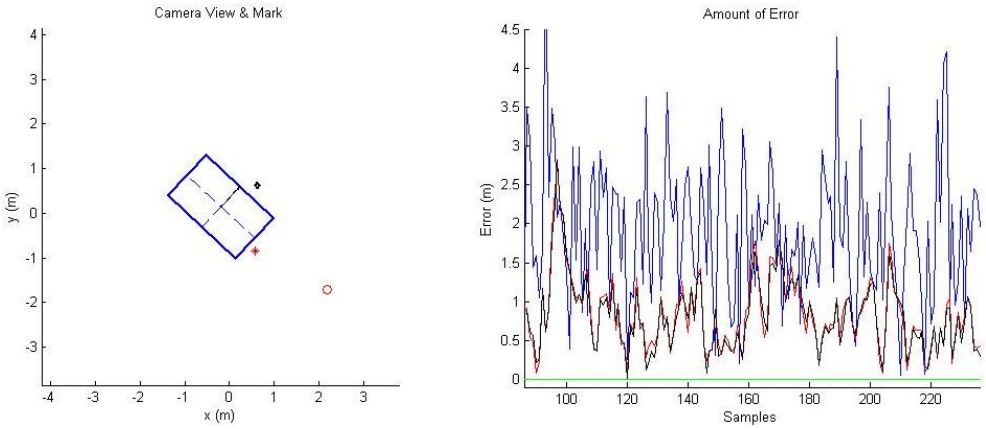


(b) Position error versus number of GPS sensors. N number of GPS sensors are assumed to be placed on the moving UGV and the distance error from the center of UAV to the improved GPS data point is calculated.

Figure 2.10.Positional error simulation with a UAV and a UGV.

While the UGV is moving, the UAV is manipulated to follow the UGV to let the UAV reference the improved position data of the UGV by detecting a mark on the UGV. In Figure 2.11(a), a blue rectangle represents the range of the UAV camera image, a thick black □ represents a mark on the UGV, a red circle represents

the pure GPS data, and a red asterisk represents the improved GPS data in ECEF coordinates using Equation 2.3. Here, Equation 2.3 collects 5 previous GPS data and averages those to get improved GPS data. In Figure 2.11(b), a blue graph indicates the pure GPS data, a red graph indicates the improved GPS data using Equation 2.3, a green graph indicates the improved GPS data using Equation 2.5, and a black graph indicates the results of the EKF by integrating the GPS and IMU sensor data on the UAV to achieve the nonlinear state estimation. Here, the green line is zero since the mark is not detected yet. When the mark is detected by the UAV, Equation 2.5 is applied and it results the improved error propagation (green graph in Figure 2.12(b)). Even if the UAV detects the UGV, it sometimes loses the UGV due to the various errors (dynamic, position, etc) and it is necessary to switch two methods in an appropriate fashion.

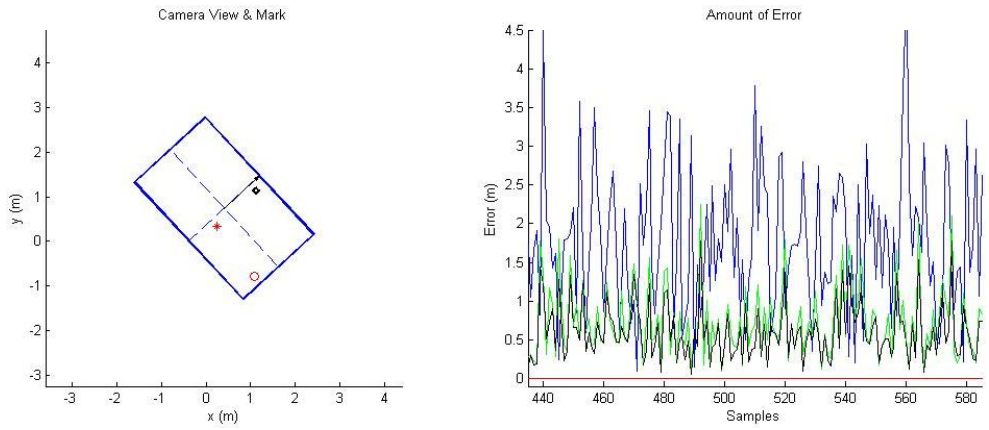


(a) A black mark is outside of the UAV camera range.

(b) Position error propagation.

Figure 2.11.Case when a mark is not detected (1 GPS on UAV and 3 GPS on UGV).

The GPS accuracy of the proposed method (Equation 2.5) is about 2.3 times better than the use of pure single GPS sensor on the UAV (Figure 2.10(b) and Table 2.1).



(a) A black mark is inside of the UAV camera range.

(b) Position error propagation.

Figure 2.12.Case when a mark is detected (1 GPS on UAV and 3 GPS on UGV).

Table 2.1Mean and standard deviation of position error of simulation results (1 GPS on UAV and 3 GPS on UGV, unit: m).

	Direct Averaging	WCL + EKF
Mean	1.7878	0.7777
Std	0.9358	0.4942

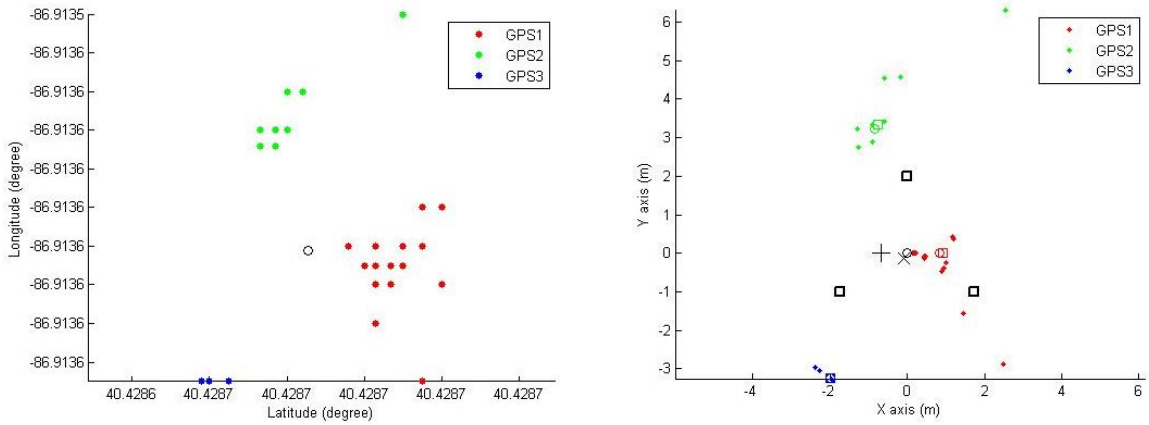
2.7 Experiment

I show here the improvement to UGV geolocation and the subsequent improvement of UAV geolocation. I give details of the experimental setup in Appendix A.

2.7.1 UGV GPS Improvement

To verify the WCL method (Equation 2.4), GPS data are collected (Figure 2.13(a)) by placing the UGV at one location $(40.427884°, -86.912540°, 190m)$ and the pure GPS data (Figure 2.13(a)) are converted into the ECEF coordinates (Figure 2.13(b)). The results show that the distance from a black $\bigcirc$ to a black $+$ is calculated as $0.6687m$ and distance from a black $\bigcirc$ to a black $\times$ as $0.1571m$ (please reference Figure 2.9 for definition of $\bigcirc$, $\times$, and $+$). This demonstrates that the usage of three

GPS sensors with the WCL method indeed improves the accuracy of GPS data about 4 times.



(a) Three pure GPS data of the UGV.

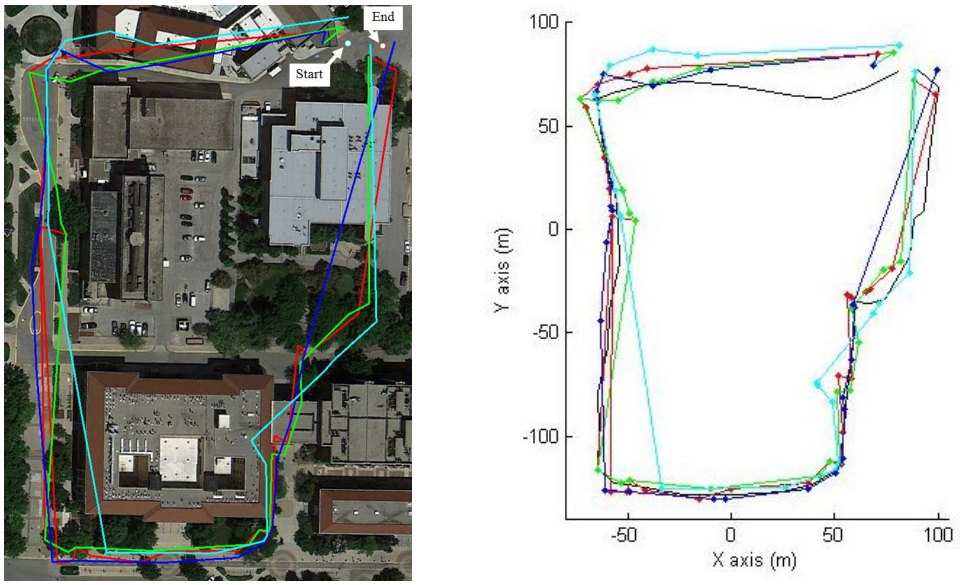
(b) GPS data converted to ECEF coordinates.

Figure 2.13.Experiment result of GPS accuracy improvement by placing three GPS sensors at one location.

2.7.2 UAV GPS Improvement

The UGV (Figure A.1(c)) is driven around the mechanical engineering building while the UAV is maneuvered to follow the UGV. Then, trajectories of the three pure GPS sensor data of the UGV and the one pure GPS sensor data of the UAV are obtained (Figure 2.14). Here, the red, green, and blue color lines represent the pure GPS sensor data on the UGV, the cyan color line represents the pure GPS sensor data on the UAV, and the black line represents the exact trajectory which the UGV drives over.

By applying Equation 2.5 and the EKF, I achieve the improved UAV trajectory (Figure 2.15(d)) and this result shows that the data of the UAV is improved about 1.6 times (Table 2.2) compare to the pure UAV GPS trajectory (Figure 2.15(c)). The large offset at the beginning and end of the trajectory (Figure 2.15(d)) is probably



(a) Pure GPS data in the Euclidean space.

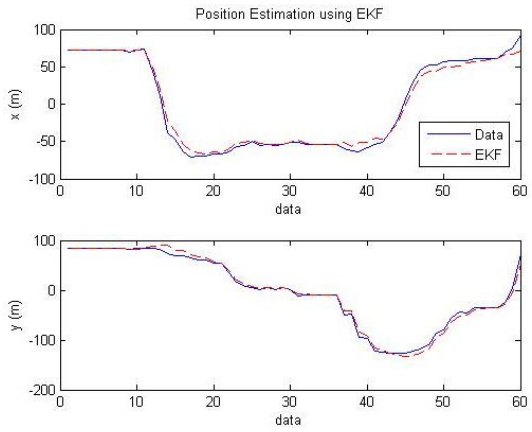
(b) Pure GPS data in ECEF coordinates.

Figure 2.14.Experiment result of pure GPS sensors on the UGV and UAV after the EKF is applied.

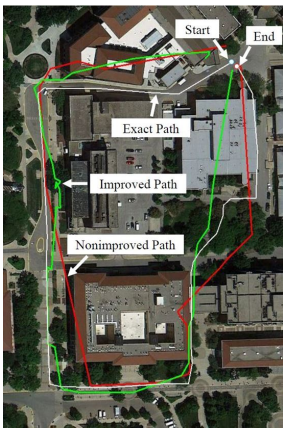
Table 2.2Mean and standard deviation of position error of experiment results (1 GPS on UAV and 3 GPS on UGV, unit: m).

	Direct Averaging	WCL + EKF
Mean	4.5216	2.8452
Std	5.3439	3.8490

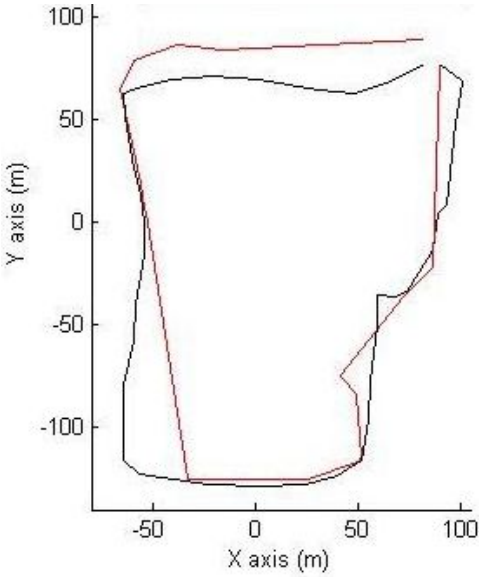
caused by the poor GPS signal communication due to tall buildings around the narrow path.



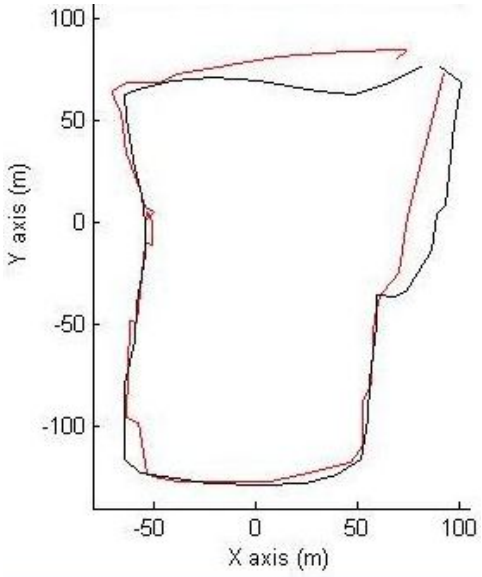
(a) EKF application to improve the UAV GPS data.



(b) Final result of the improved GPS data of the UAV.



(c) Comparison between the pure GPS data of the UAV and the exact trajectory of UGV.



(d) Comparison between the improved GPS data of the UAV and the exact trajectory of UGV.

Figure 2.15.Experiment result of pure GPS sensors on the UGV and UAV.

2.8 Concluding Remarks

I have shown that the accuracy of a GPS sensor on a UAV can be improved by referencing the integrated GPS data and a mark on the UGV by using the WCL and EKF. Using three GPS sensors on the UGV and one GPS sensor on the UAV result

about 2.3 times (simulation) and 1.59 times (experiment) better accuracy compare to the single GPS sensor on the UAV. This work can be applied to the UAV swarming formation control with increased number of UGVs as well. My work opens up several possibilities for both system and algorithm developments:

1. The optimal patterns on the UGV needs to be constructed so the pattern size can be minimized and can handle dynamic tilting of the UAV. Gusty situations can be handled within the present framework because of the large frame rate ($60fps$) of the camera, but the limiting factor is the number of pixels on the camera's image.

2. A method of detecting the UGV even with much longer distance needs to be developed to avoid the loss of detection and limitation of application time zone since the local weather conditions or obstacles obscure the mark on the UGV and image based UGV detection limits the usage of this work only during day time. Combining my method with the use of local terrain landmarks may accomplish this result.

CHAPTER 3: ORIENTATION SENSING

Orientation of UAVs with only one magnetic compass can be compensated with another orientation calculated by solar compass using the Three Pixel Theorem (TPT) when UAVs fly around areas with large magnetic disturbances. Both indoor and outdoor experiments prove that the solar compass works much better than the magnetic compass in the aspect of linearity of the data.

3.1 Background and Motivation

Historically, for the classical celestial navigation methods, some great inventions have to be mentioned. Sextants [67] and Astrolabes [68] were invented for measuring the angle of a celestial body above the horizon and calculating the desired location and direction information differentially. In modern life, I require much more accurate and stable navigation methods since flying objects, such as UAV is getting being a necessary item which a nation, a company, and a person should possess in order to utilize in a variety of fields. Having a precise controllability of UAV will certainly revolutionize the life of modern human in the aspect of tremendous applications of UAV. To achieve fully autonomous UAV, it should be able to precisely fly over assigned paths to perform various missions which is guaranteed by having precise system models, reliable orientation, and position estimation of UAV. Unfortunately, none of these three categories are promised during UAV flights due to heavily disturbed magnetometer sensor and GPS sensor. These issues are attempted to be solved by using magnetic compass integrated with solar compass.

Section 3.2 explains overall methods of using azimuth, zenith, and sun vector to invent a solar commpass. Section 3.3 shows indoor and outdoor experiment results

of both the magnetometer and solar compasses. Section 3.4 contains the conclusion and future works.

3.2 Magnetic Compass Integrated with Solar Compass

Most of vessels and planes use means of compasses, GPS compass; fluxgate compass, electro-magnetic resistors, and gyro compass, which work independently to determine direction and heading to avoid some erroneous data flows under unexpected situation. In a similar way, orientation of ARdrone is compensated with solar compass. Although the earth's magnetic field can provide a magnetic compass for determining orientations, it may be susceptible to underground iron bearing minerals or other electronic device if there is no any appropriate electromagnetic protection. Many researches have been done to control a quadrotor, but it does not work well in urban environment where full of magnetic disturbance exist. Without knowing accurate orientation, it will be much harder for a quadrotor to track given paths. So, magnetic compass integrated with solar compass is proposed to be used to determine orientation of UAV when local weather condition is sunny. In fact, spherical sundials were built in medieval times for precise calibration of the calendar, and the determination of latitude and longitude. These were large structures that provided great precision for their times. In my work, I have aimed to replicate their accuracy with smaller components with dynamic platforms in mind. To achieve this goal, I designed and built an innovative hemispherical structure with light sensors arranged in an array of different positions. As the sun, moon and stars are far away from us, their light can be seen as parallel rays from a source. Depending upon the orientation of each sensing element, it receives a different intensity of incident light. Using signals from at least three sensors, I can obtain the relative orientation to the sun in terms of the Azimuth and Zenith angles.

From the observer's perspective, the sun's relative orientation can be determined by two angles Azimuth and Zenith as shown in Figure 3.1. Azimuth is defined as

a horizontal angle measured clockwise from any fixed reference plane. Zenith is the direction pointing directly above a particular location. Since a sun vector is unique for a specific location at specific time, so it is possible to use an accurate clock and location information (Latitude and Longitude from GPS) to determine a specific sun vector. Time and location or the sun-vector, either of them can be mathematically calculated given the other. The astronomical knowledge needed to do this has been available for centuries. In this work, I perform these computations in real time using widely known algorithms [69, 70]. Most of these algorithms have achieved uncertainties of $\pm 0.01°$, even with $\pm 0.0003°$ in calculating solar Zenith and Azimuth angles. Figure 3.2 shows a MATLAB® interface for calculating and showing the sun vector. The north direction is zero for Azimuth and clockwise is positive. A direction which is vertical to the level is 90 degree for Zenith. For instance, Figure 3.2 shows a particular location at Purdue University has a sun vector of about $[\text{Azimuth}, \text{Zenith}] = [214.7°, 42.5°]$ at the time of March 16^{th}, 2012, $14:35:21(UTC-5)$. Since I intend to build a solar compass, only Azimuth angle of the calculated sun vector is needed. Once a standard Azimuth angle of a sun vector is defined, it can be compared with the sun's Azimuth angle from the UAV's perspective. The difference between these two angles indicates an orientation which I can use as a compass. For example, if a standard Azimuth angle of the sun is 150° clockwise to the north and the observer gets an Azimuth angle of a sun vector which is 30° clockwise from his facing direction, then this indicates that observer is $(150° - 30°) = 120°$ clockwise to the north.

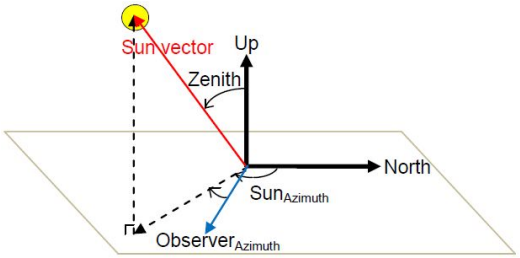


Figure 3.1.Azimuth, Zenith, and Sun vector.

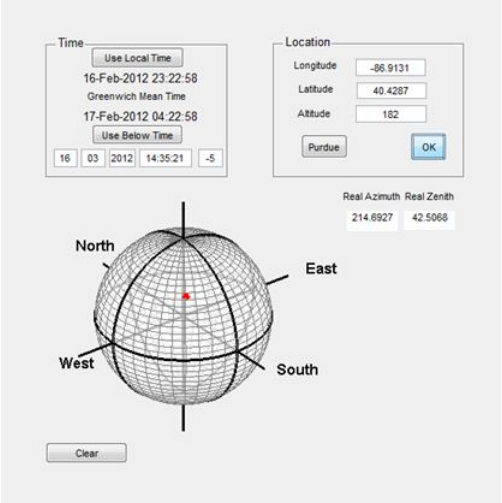


Figure 3.2.GUI for sun vector calculation.

To calculate the sun vector from UAV's perspective, Three Pixel Theorem (TPT) [71] can be uses. According to TPT, the sun vector can be determined by drawing a sphere inside the hemisphere using three pixel positions and the origin of the hemisphere as shown in Figure 3.3 and the sun vector, $\vec{r_0}$, can be derived as Equation 3.1,

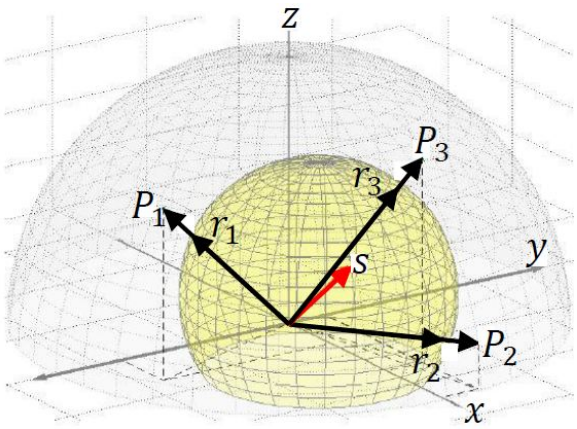


Figure 3.3.Hemispherical Solar Sensor (HSS) pixel coordinates and orientation vector components.

$$\vec{r_0} = \frac{1}{2}(A^T A)^{-1}(A^T)\vec{b}, \tag{3.1}$$

where $\overrightarrow{r_0}$ which is the sun vector($\in R^3$), A, and b are defined as

$$A = \begin{pmatrix} x_1 - x_2 & y_1 - y_2 & z_1 - z_2 \\ x_2 - x_3 & y_2 - y_3 & z_2 - z_3 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\ x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix}, \quad \vec{b} = \begin{pmatrix} x_1^2 + y_1^2 + z_1^2 - x_2^2 - y_2^2 - z_2^2 \\ x_2^2 + y_2^2 + z_2^2 - x_3^2 - y_3^2 - z_3^2 \\ x_3^2 + y_3^2 + z_3^2 - x_1^2 - y_1^2 - z_1^2 \\ x_1^2 + y_1^2 + z_1^2 \\ x_2^2 + y_2^2 + z_2^2 \\ x_3^2 + y_3^2 + z_3^2 \end{pmatrix}.$$

Then I can calculate orientation of ARdrone by using r_0 as shown in Figure 3.4 and Equation 3.2.

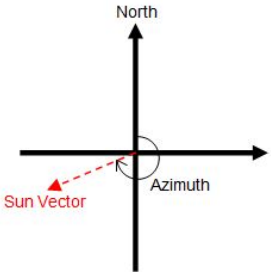


Figure 3.4.Azimuth angle calculation.

$$\begin{aligned} \text{Azimuth angle } (^\circ) &= cos^{-1}\left(\frac{|y_0|}{\sqrt{x_0^2 + y_0^2}}\right), && (x_0, y_0 \geq 0), \\ &= cos^{-1}\left(\frac{|y_0|}{\sqrt{x_0^2 + y_0^2}}\right) + 90^\circ, && (x_0 \geq 0, y_0 < 0), \\ &= cos^{-1}\left(\frac{|y_0|}{\sqrt{x_0^2 + y_0^2}}\right) + 180^\circ, && (x_0 < 0, y_0 < 0), \\ &= cos^{-1}\left(\frac{|y_0|}{\sqrt{x_0^2 + y_0^2}}\right) + 270^\circ, && (x_0 < 0, y_0 \geq 0). \end{aligned} \tag{3.2}$$

3.3 Experiment

Least Square Estimation (LSE) method is applied to real application using a hemispherical array of 9 light intensity sensors (CdS photoconductive cells) as shown in Figure 3.5 attached on the ARdrone to obtain a sun's Azimuth angle from UAV's perspective as shown in Figure 3.6. When those light sensors are under the sunshine, each sensor receives different amount of light depend on their position vectors and a sun vector, and this information is used for calculating the sun vector.

3.3.1 Indoor Experiment

To do experiment inside the lab, a desk lamp is used as the fake sun which is set to direct South as shown in Figure 3.7(a). With this experiment setup, Azimuth can be calculated as shown in Figure 3.7(b).

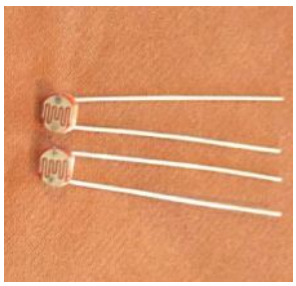
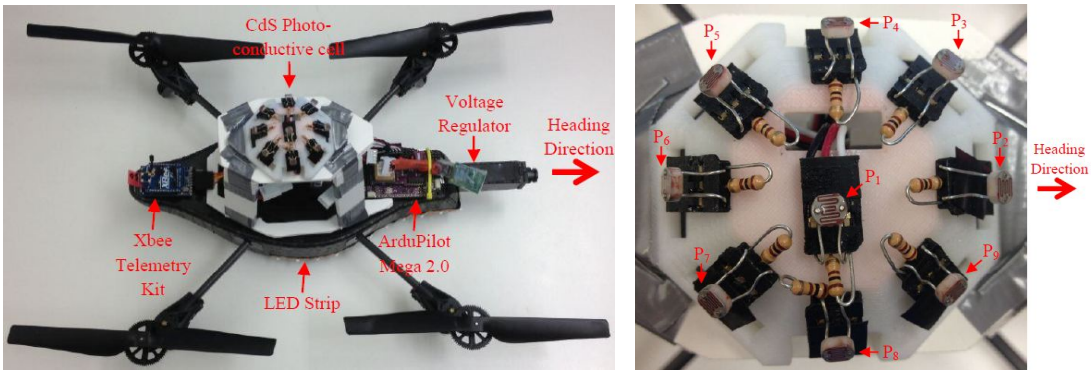


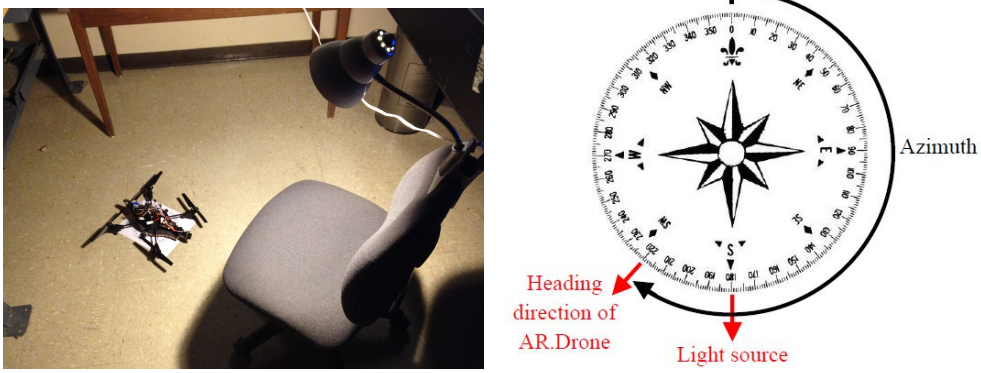
Figure 3.5.CdS photoconductive cells.

Experiment results are shown in Table 3.1 and Figure 3.8. This experiment is performed by spinning ARdrone from 0° to 360° with 30° step. According to Figure 3.8, solar compass is more robust than the magnetic compass in the aspect of straightness. Magnetic compass is quite unreliable due to unknown magnetic field disturbance since the experiments are performed in lab where various electronic devices are located.



(a) Overall system of ARdrone. (b) 9 hemispherical light sensors set up.

Figure 3.6.CdS photoconductive cells fusion on ARdrone.



(a) ARdrone on a compass drawn on a white paper with a fake sun. (b) Azimuth calculation.

Figure 3.7.Solar compass test setup.

3.3.2 Outdoor Experiment

Outdoor experiment is performed by flying ARdrone straightly for about $20m$ at Squirrel Park located at the south west of Purdue University as shown in Figure 3.9(a,b). The location of the Squirrel Park is [longitude, latitude, altitude] = [-86.931821°, 40.423450°, $188.14m$].

Table 3.1Comparison results of the solar and magnetic compasses (unit: degree, MSE: Mean Square Error).

Expected Azimuth	Solar compass	MSE (Solar)	Magnetic Compass	MSE (Magnetic)
0	0.05	0	41.3	1702.4
30	45	225	58.7	824.3
60	67.5	56.3	72.9	166.4
90	112.5	506.3	88.4	2.7
120	108.5	131.3	104.5	241.2
150	154.7	22	123.2	720.4
180	183	8.8	177.1	8.6
210	232.1	486.2	182.5	756.8
240	263.2	537.8	297	3250.1
270	292.5	506.3	343.9	5458.3
300	300.7	0.5	5.2	4247.1
330	315	225	24.3	2946.3
		225.4		1693.7

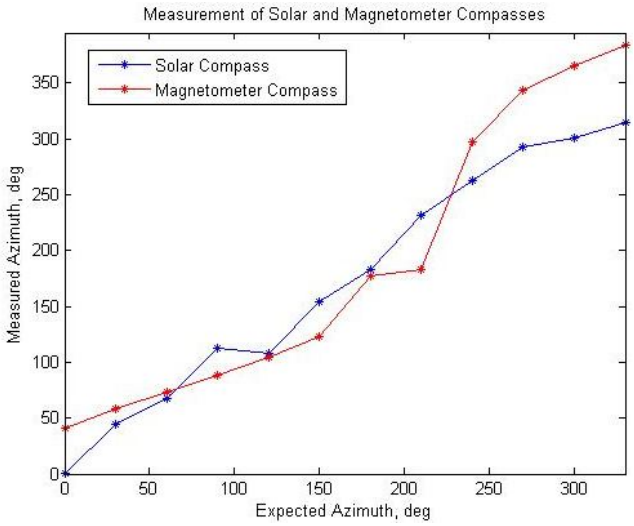
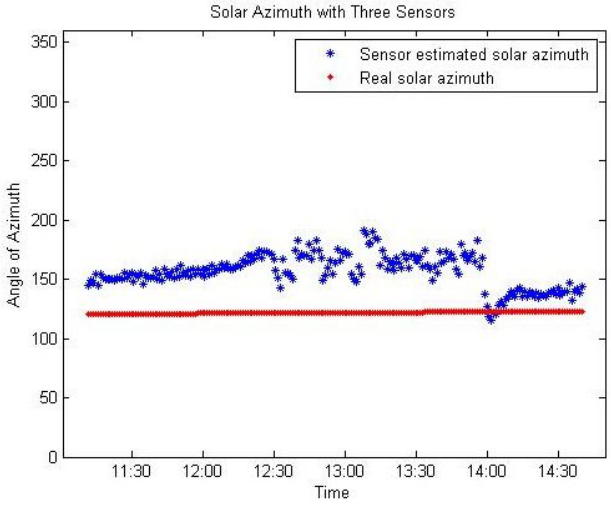


Figure 3.8.Azimuth comparison between solar and magnetic compass.

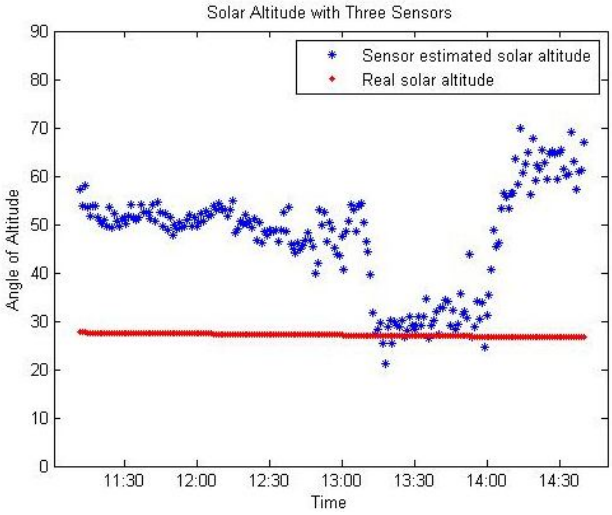
(a) Outdoor test 1.

(b) Outdoor test 2.

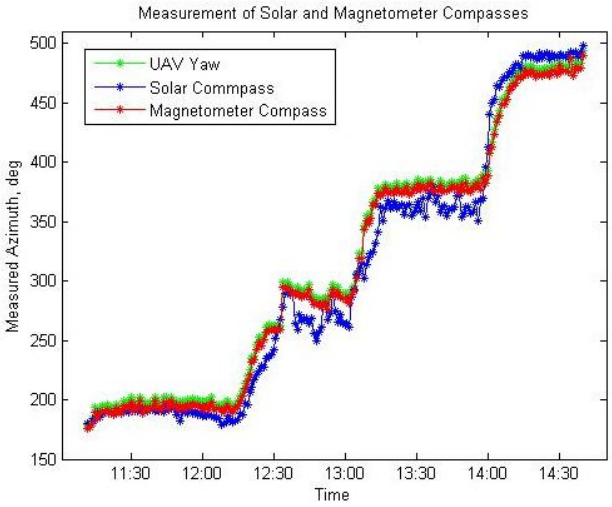
Figure 3.9.UAV is flying straightly along the green line on the ground for $20m$.



(a) Outdoor test result 1.



(b) Outdoor test result 2.



(c) Outdoor test result 3.

Figure 3.10.Three outdoor test results between magnetic compass and solar compasses.

3.4 Concluding Remarks

Two methods, solar compass and magnetic compass, to improve the orientation of UAV were studied. Solar compass is designed and compared with Magnetic compass which brought conclusion that Solar compass works much better than Magnetic compass when good weather condition is allowed according to both the indoor and outdoor experiments. Although solar compass worked better than magnetic compass at this time, this will not be the case all the time due to the whether conditions. Therefore, logical algorithms will be needed to integrate these two compasses together. As I have seen in the experiments, relying only on a magnetic compass could bring wrong maneuver of UAV, so it should be avoided as much as possible. Applications of my hemispherical solar sensor are broad and flexible. It can be used in solar electricity generation field which can help the solar panels point to the sun and receive maximum sun light.

CHAPTER 4: UAV AVAILABILITY

I develop a wireless, contact free power transfer mechanism that is safer than the direct metallic contact and robust to imperfect alignment on landing at the base station. A magnetic field is created using inductors on both the transmitting and receiving sides. I use the inductive wireless recharging to increase autonomy and decrease the sensor interference by reducing the inductor loop size. By locating four independent small receiver loops and corresponding four circuits around the quad-rotor UAV, I can increase safety from circuit malfunctions in comparison to the use of just one loop. In addition, more loops permit larger current flow which brings more efficient power transfers. On the base station, four folding robotic bars are used to realign the receiver loops over the transmitter loops. After adequate recharging as measured by battery voltages or power consumption at the base station, the UAV sends a signal to the base station to open the robotic bars to fly away.

4.1 Background and Motivation

Small Unmanned Aerial Vehicles (UAVs) promise significant performance and safety improvements to routine tasks such as inspection, surveillance, and policing, but limited battery power constrains the extent and autonomy of these tasks at present. As of today, batteries are the major problem for UAVs to travel long distances. At least half of the energy in the battery must be saved to travel back to the launch site for recharging which is again time-consuming. In general, there are two methods to provide a fully charged battery to a UAV; recharging or replacing on site. The concept of replacing the battery was previously performed [72–74], but this method is not preferred since the battery replacing mechanism would have to be different in each case depending on the type of UAV which is very inefficient.

In addition to the final choice of the recharging method, the actual method of how to recharge the battery is a big issue. Due to safety concerns, metal contacts to the UAV [75] might be a bad choice depending on weather conditions (rain or snow). Also, it assumed that UAVs can land exactly on the ground station with the help of a Vicon camera system which gives the exact position data of both the UAV and the ground station, but I should acknowledge that the Vicon camera system cannot be used outside and instead GPS sensors should be used by taking into account some intrinsic position errors. In this case, wireless power transfer is more appropriate since it is safer and manageable for misaligned landing and one possible method to transfer power to the UAV is using the inductive wireless power transfer. This method can be established without any direct contact between the ground station and the UAV. Since it can transfer energy as long as the receiver loops and transmitter loops are located within a certain boundary, so I have a higher freedom of maneuvering the UAV to land on the right spot. In addition, although one big inductive loop can be attached to the UAV by encircling it, such a loop around the UAV may interfere with other electronic components of the UAV due to the appearance of heavy magnetic fields. Using several small induction loops around the UAV can solve the problem instead.

In general, there are three wireless recharging methods; laser powering, Radio Frequency (RF) powering, and inductive couple powering. Laser powering method was invented half a century ago by William C. Brown [76, 77] and has been recently applied to real life thanks to powerful, efficient and inexpensive laser diodes. However, it has a serious problem regarding safety of human eyes even though extensive research and experiments have been conducted [57–59]. RF powering has a wider range of working area but it is very inefficient in the close distance compared to inductive coupling since a ground station needs a more sophisticated design due to Federal Communications Commission (FCC) regulation. The regulation states that the RF range of frequency must be within the enclosed area for safety concerns with cellular phone. In that sense, the Inductive coupling method has more advantages for UAV

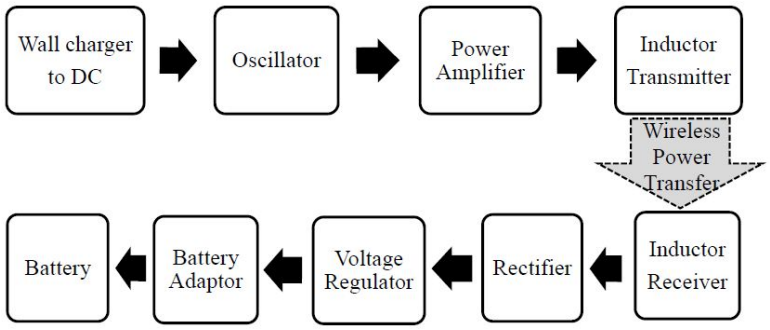
autonomous operations the reason being that it is simpler, cheaper, and an efficient way to recharge batteries on UAVs.

Section 4.2 explains the design of the wireless charging system using the inductors. Section 4.3 describes the autolanding system design for the UAV and section 4.4 explains the design of the Unmanned Ground Station (UGS). Section 4.5 shows the calculation of total system efficiency. Section 4.6 shows the experimental results and section 4.7 contains the conclusion of this chapter.

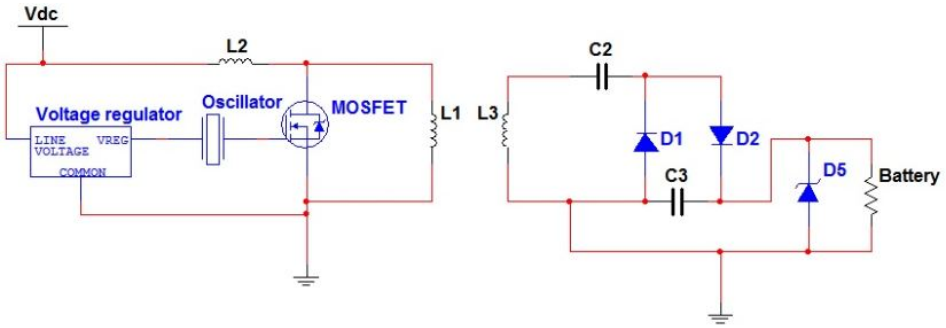
4.2 Transmitter and Receiver Loop Design

Most of the time, researchers use a single pair of transmitter and receiver inductors for wireless power transfer for UAVs [78, 79]. However, using several small loops can ensure safe wireless power transfer with four independent power receiving circuits compared to the single circuit from the circuit malfunction. Beside, these smaller loops create less interference with other sensors on board by distributing magnetic fields.

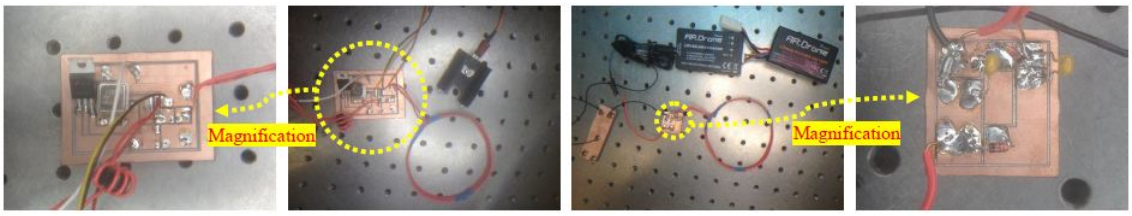
The inductive coupling system (Figure 4.1) consists of two main parts; the transmitter side and the receiver side. Transmitter side contains a power source, an oscillator, a power amplifier, and an inductor. At the receiving side, there are a receiving inductor, a rectifier, a voltage regulator or a limiter, a battery adapter, and a battery on the UAV. On the transmitter side, DC power source from walls is converted into an AC signal using an oscillator where the power from wall charger can be replaced by wind turbines and/or solar panels. The AC signal is amplified by the power amplifier to increase power, then the AC signal is applied to the transmitting inductor. On the receiver side, the receiver inductor which is the same size with the transmitter inductor takes current from transmitter side and converts back to DC with a Villiard voltage doubler. Since the rectified voltage is too high, a zener diode is used to reduce the input voltage. Here, the zener diode is used to reduce the power consumption compared to the voltage regulator.



(a) Schematic diagram.



(b) Circuit diagram.



(c) Hardware setup.

Figure 4.1.Transmitter and receiver of the wireless charging system.

Inductive coupling loses most of its energy to the air during the wireless power transfer process, so the efficiency decreases significantly as the distance between the transmitting and receiving loops increases. Therefore it is essential to place two inductors closely and to have an ideal alignment to achieve the highest efficiency.

The following sections for transmitter and receiver loop design are done with the following design constraints; 4 pairs of transmitter and receiver loops are used; a LiPo battery has properties of $1000mAh$, $11.1V$, and maximum charging rate of $1A$; radius

of each loop is $4cm$; number of turns of each loop is 6; total weight of loops should be less than $250g$.

4.2.1 Inductance

Most small size UAVs have limited allowable space and weight load to attach inductors around the body. The radius of loop is predetermined as $4cm$ to satisfy space allowance around the UAV. The number of turns of the receiving loop is set to be 6 as additional design constraint. With the given conditions, the inductance, L, can be calculated as [79],

$$L = \mu_0 r N^2 \left(\ln \frac{8r}{c} - 2 + Y \right), \tag{4.1}$$

where μ_0 is a magnetic constant (4π x $10^{-7} H/m$), r is loop radius (m), N is a number of turns of loop, c is wire radius (m), and $Y \in \{0, 0.25\}$. Y is 0 when the current is uniformly distributed over the surface of the wire (skin effect) and is 0.25 when the current is uniformly distributed over the cross section of the wire [80]. So, the loop has L value as,

$$L = 4\pi 10^{-7} \cdot 0.04 \cdot 6^2 \left(\ln \frac{8 \cdot 0.04}{0.00075} - 1.75 \right) = 7.8\mu H. \tag{4.2}$$

However, simulation using ANSYS HFSS shows that inductance changes as frequency increases (Figure 4.2(a)) and the inductance when frequency is $12MHz$ is about $5\mu H$. $12MHz$ is selected for safety issue from possible sparkling since sparkling minimizes at high frequencies. In contrast to the inductance values from theoretical (Equation 4.2) and simulation (Figure 4.2(a)) results, $6.6\mu H$ is measured by experiments. The difference might be come from the PVC cover around the wire which blocks the wireless power transfer.

4.2.2 Mutual Inductance

Mutual inductance, M, determines how much electromagnetic field is created by the primary loop. The transmitting loop produces an electromagnetic field that inducts current flows to the receiving loop with alternating current. If the mutual inductance becomes large, then the coupling coefficient becomes large as well. This results in higher efficiency between the loops. Mutual inductance is defined as [81],

$$M = \frac{\mu_0 N_1 N_2 A}{l}, \tag{4.3}$$

where N_1 is a number of turns in loop 1, N_2 is a number of turns in loop 2, A is a cross sectional area (m^2), and l is a loop length/width of loop in total (m). Mutual inductance is affected greatly by the vertical distance between the loops (Figure 4.2(b)). Since the mutual inductance drastically drops after about $1cm$ distance, $1cm$ gap is used as the maximum gap between two loops to design the locations of the transmitter loop on the UGS and the receiver loop on the UAV.

4.2.3 Transmitter Loop

An AC signal generator is required to make AC from DC for the inductor. To make an AC signal, crystal oscillator is chosen. IRF510 MOSFET ($0.96 [82]) is selected for the power amplifier since commercial power amplifiers are too expensive. To achieve a current close to $1A$, about $30V$ with $1A$ is required to IRF510 MOSFET. It is better to use IRF510 MOSFET rather than IRFZ44 since IRF510 MOSFET operates at lower temperature. A heat sink is attached to MOSFET to ensure a working temperature.

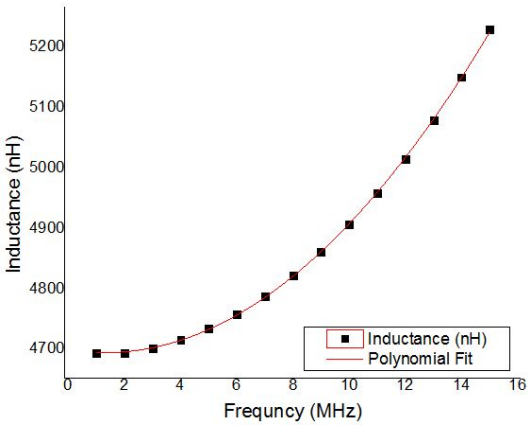
4.2.4 Receiver Loop

AC signal needs to be converted back to DC to charge the battery and a Villiard voltage doubler which is composed with 1N4148 is used to rectify the AC to DC.

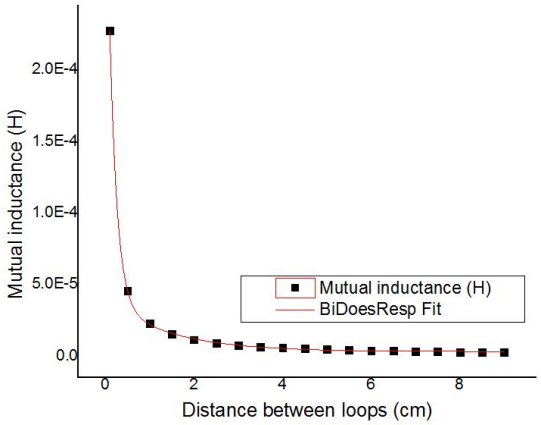
Then, a Zener diode (1N5248) is used to replace a voltage limiter at $18V$ which is required to charge the UAV battery. I choose a Zener diode since it operates with smaller power compared to a voltage regulator chip and adds simplicity to the circuit design. The receiving loop is designed to be identical to the transmitting loop due to space and weight load limit of the UAV. I plot 20 data of the vertical distances (Figure 4.2(c)) and horizontal distances (Figure 4.2(d)) and corresponding voltage changes. To see the effect of neighbor transmitter loops as a receiver loop horizontally moves, two transmitters are put with $24cm$ gap (Figure 4.3(a)). Results show that receiving voltage drops drastically when vertical distance is $2.5cm$ and horizontal distance is $2cm$. The slight increase at the horizontal distance of $8cm$ happens when the boundary of receiver loop is directly above the boundary of transmitter loop. Also, since the loops are horizontally aligned with $24cm$ distance, I need to study the power transfer rate at the middle point. At $12cm$ of horizontal distance, $0V$ is measured and it demonstrates that none of the loops are electromagnetically overlapped. So, I can assume that the other three transmitter loops do not transfer power to the receiver what I am looking at.

4.2.5 Loop Efficiency

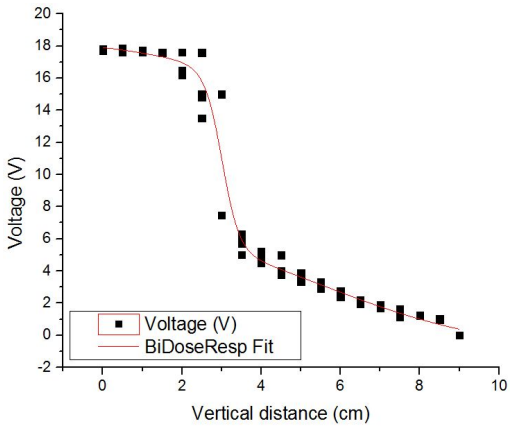
When $27.62V$ and $0.957A$ $(26.43W)$ is supplied on the transmitter side, $176V$ with $0.02A$ $(3.52W)$ is detected on the receiver side with a resistance of 50Ω. So, the loop efficiency is determined by comparing the input power to the output power; primary to secondary loop has 13.32% efficiency; total system has 7.85%. System efficiency of 7.85% is measured with a 50Ω resistor after the rectifier and voltage limiter. The main reason of efficiency drop is due to the weakly coupled inductors. The majority of energy loss within the designed circuit takes place between the two loops and this problem can be complemented by putting a ferromagnetic core in the transmitter side to increase the magnetic field through the air and UGS acrylic cover.



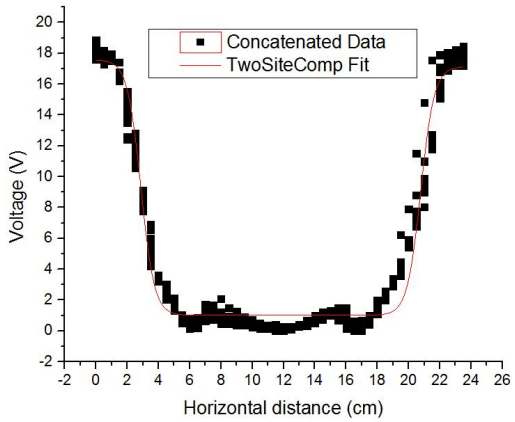
(a) Frequency Vs. Inductance.



(b) Vertical distance between loops Vs. Mutual inductance.



(c) Vertical distance and corresponding voltage changes.



(d) Horizontal distance and corresponding voltage changes.

Figure 4.2.Change of inductance and mutual inductance and voltage change corresponding to vertical and horizontal distance change.

4.3 UAV Autolanding Design

4.3.1 Position Controller Design

On reaching the destination, the UAV flies around the area according to a spiral pattern. When the landing mark is detected, the UAV stops and hovers over it using simple PID controller [83] to minimize the distance between the center of image frame and the center of detected mark. The distance between the UAV and UGS ($L2$ in Figure 4.3(a)) can be calculated using $L_2 = \frac{l_2 L_1}{l_1}$ where l_1 is an image length of the

side of the square mark, l_2 is an image length from the image center to UGV center, and L_1 is a real length of the side of the square mark. Then, l_2 can be represented as,

$$l_2 = \| \frac{1}{2}[I_x, I_y], \frac{1}{4}\sum_{i=1}^{4} c_i \|, \tag{4.4}$$

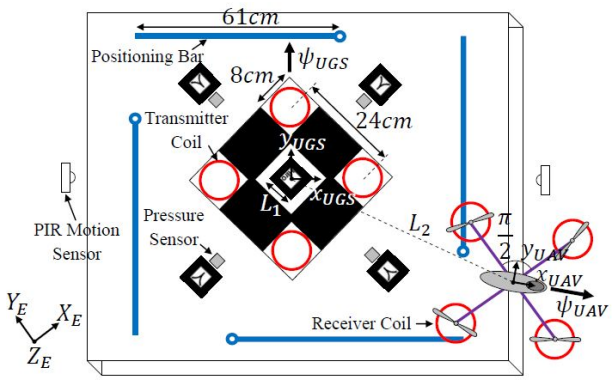
where I_x is a width of image frame, I_y is a height of image frame, and c_i is the position coordinate of each corner of the mark. So the real distance between the UAV and UGS becomes

$$L_2 = \frac{L_1 \| \frac{1}{2}[I_x, I_y], \frac{1}{4}\sum_{i=1}^{4} c_i \|}{l_1}. \tag{4.5}$$

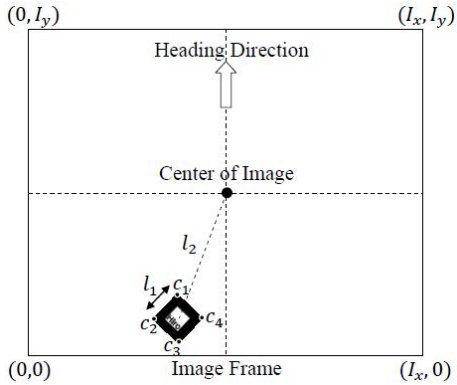
Then, when L_2 becomes less than or equal to a specific threshold, it means the UAV is hovering right above the center of UGS. To filter out any possible image processing error, the mark needs to stay detected for a certain number of iterations. If not, the detected mark is declared as an error and the automatic landing algorithm restarts.

4.3.2 Attitude Controller Design

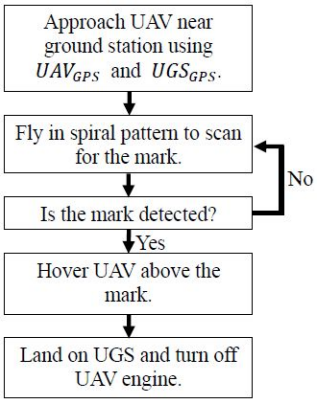
While the position controller maneuvers the UAV to the center of UGS, the attitude controller needs to align inductors between receiver loops and transmitter loops. Since 3-axis magnetometers are used both the UAV and UGS, I know both the UAV yaw angle (ψ_{UAV}) and the UGS yaw angle (ψ_{UGS}). So, the amount of rotation for the UAV ($| \psi_r |$) and the direction of rotation ($\angle\psi_r$) can be determined as



(a) UAV hovering above the UGS.



(b) Detected mark located within the UAV camera view.



(c) Flow chart of the UAV control algorithm.

Figure 4.3.UAV autonomous operation.

$$
|\psi_r| = min \begin{bmatrix} |\psi_1| \\ |\psi_2| \\ |\psi_3| \\ |\psi_4| \end{bmatrix} = \begin{bmatrix} |\psi_{UGS} - (\psi_{UAV} + \frac{\pi}{4})| \\ |\psi_{UGS} + \frac{\pi}{2} - (\psi_{UAV} + \frac{\pi}{4})| \\ |\psi_{UGS} + \pi - (\psi_{UAV} + \frac{\pi}{4})| \\ |\psi_{UGS} + \frac{3\pi}{2} - (\psi_{UAV} + \frac{\pi}{4})| \end{bmatrix}, \tag{4.6}
$$

$$
\angle\psi_r = \begin{cases} \text{CCW} & \text{if} \quad \psi_i > 0, \\ \text{CW} & \text{otherwise.} \end{cases}
$$

When the position error (e_{pos}) and the yaw angle error (e_{yaw}) become less than certain thresholds, the UAV starts to descend and lands on the UGS.

4.4 UGS System Design

MediaTek MT3329 GPS sensor is used to send the location of UGS to UAV, but this sensor has a horizontal position accuracy as low as $3.0m$ Circular Error Probable (CEP) (50%). To compensate this, multiple GPS sensors are used to increase position accuracy by locating at L_{bar} distance ($2m$) away from the center of UGS (Figure 4.4) with the concept of Weighted Centroid Localization (WCL) as [55],

$$
\begin{aligned}
P_i'' &= \overline{P} + \frac{1}{m}\left(P_i' + \sum_{j=1, i\neq j}^{m} R_z(\theta) P_j'\right), \qquad (4.7)\\
P_i' &= \frac{\sum_{k=1}^{n} w_{i,k} P_{i,k}}{\sum_{k=1}^{n} w_{i,k}} - \overline{P},\\
P_j' &= \frac{\sum_{k=1}^{n} w_{j,k} P_{j,k}}{\sum_{k=1}^{n} w_{j,k}} - \overline{P},\\
\overline{P} &= \frac{1}{mn}\sum_{i=1}^{m}\sum_{k=1}^{n} P_{i,k},
\end{aligned}
$$

where P_i is the ith GPS data, m is the total number of GPS sensors, n is the total number of data collected at one position, $R_z(\theta)$ is the rotation matrix where θ can be calculated as $\theta = \frac{2\pi}{m}$, $w_{i,k}$ is the weight of ith GPS sensor at data sample k where k can be calculated as $k = \frac{1}{\left(d_{i,k}\right)^g}$, $d_{i,k}$ is the distance between a center of enough number to data to the kth data of ith GPS sensor, and g is a degree.

When the degree g is set to be zero, WCL becomes Centroid Localization (CL). Estimated locations of the other GPS sensors except the reference GPS sensor are moved using rotation matrix about $\overline{P}$ to near the reference GPS sensor, and then

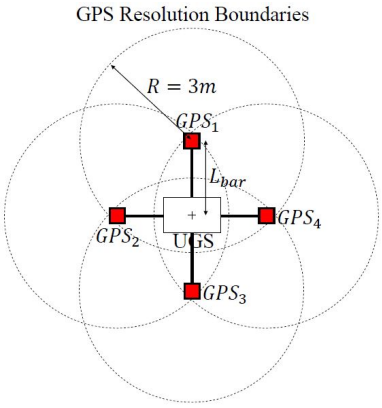


Figure 4.4.UGS with multiple GPS sensors.

I average those to calculate the improved location of the reference GPS sensor. In addition, past q number of GPS data are used to improve the current GPS data and this method is named as Receding Horizon Collection (RHC). The RHC is described as,

$$P_{UGS} = \frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{q}\sum_{k=n-(q-1)}^{n} P'''_{i,k}\right), \tag{4.8}$$

where q is the number of past GPS data considered for RHC and n is an integer greater than or equal with q. If all equations are combined, P_{UGS} becomes

$$P_{UGS} = \frac{1}{m^2 q}\sum_{i=1}^{m}\sum_{k=n-q+1}^{n}\left(P'_i + \sum_{j=1,i\neq j}^{m} R_z(\theta)P'_j\right)_k. \tag{4.9}$$

Even if the UAV comes near the UGS using its own and UGS GPS data, the UAV is not able to precisely land on UGS due to the inherent errors of GPS sensors. Therefore, the UAV should have a method to detect the exact location of the UGS and land on it without any collisions. One solution for this problem is to use marks as indications of the UGS. An example is using the NyARToolkit [60]. With the help of the NyARToolkit algorithm, black square marks on the UGS can be detected visually and the positions of the four corners of the square mark can be obtained which are used to guide the UAV. Once the UAV detects the mark, the UAV lands on the UGS.

The landing of UAV can be detected using two different types of sensors. Two Passive InfraRed (PIR) motion sensors and four Force Sensitive Resistor (FSR) sensors are placed on the UGS to detect the landing of the UAV. The FSR sensor used can sense weights in the range of $100g$ to $10kg$.

For smooth movements of the UAV and UGS robotic bars, Pololu^{TM} ball casters are used at each of UAV legs and UGS robotic bars. To choose appropriate type of servo motors for the movements of four robotic bars, I need to consider the stall torque of servo motors and the friction force of ball casters attached on the UAV and UGS bars. Commercial servo motors which are widely used with Arduino microcontroller have a feature of stall torque with an unit of $kg \cdot cm$. With this stall torque (T_s), stall force (F_s), and a robotic bar length (l_{bar}), I can represent stall torque as $T_s = gx$ where g is the gravitational acceleration ($9.807m/s^2$) and x is the servo torque ($kg{\cdot}m$). Then, the stall force applying at the end of the robotic bar (Figure 4.5) is

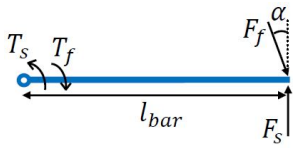


Figure 4.5.Calculation of required torque for a robotic bar.

$$T_s = l_{bar} F_s = gx, \tag{4.10}$$

and it brings

$$F_s = \frac{g}{l_{bar}} x. \tag{4.11}$$

In Equation 4.11, the total robotic bar length is used since the maximum torque occurs at the end of robotic bar. To move the UAV, I should satisfy $F_s > F_f$, so

$$\frac{g}{l_{bar}} x > \max\{F_{f,uav} \cos\alpha + F_{f,bar}\}. \tag{4.12}$$

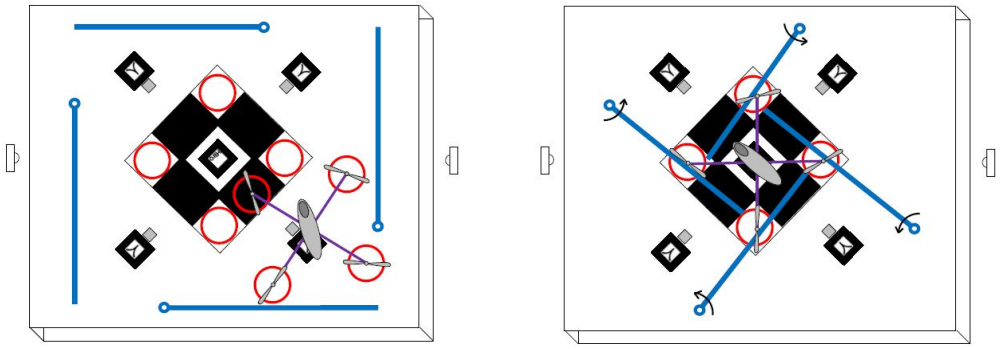
With the frictions of ball casters μ_{bar} and μ_{uav}, I can derive

$$x > l_{bar}\max\{\mu_{uav}m_{uav}\cos\alpha + \mu_{bar}m_{bar}\}. \tag{4.13}$$

By assuming that $\mu = \mu_{uav} = \mu_{bar}$ and the fact that the maximum friction occurs when α is zero, the above equation becomes

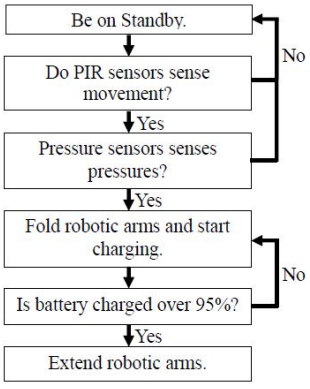
$$x > l_{bar}\mu(m_{uav} + m_{bar}). \tag{4.14}$$

By substituting the measured data ($l_{bar} = 0.61m$, $\mu = 0.5$, $m_{uav} = 0.536kg$, and $m_{bar} = 0.025$kg), Equation 4.14 becomes $x > 0.1711kg \cdot m$. Therefore, I need to choose servo motors which has larger than $17.11kg \cdot cm$ output torque. The UGS operation is shown in Figure 4.6.



(a) Initial position of the UGS.

(b) Final position of the UGS.



(c) Flow chart of the UGS control algorithm.

Figure 4.6.UGS autonomous operation.

4.5 Total System Efficiency

The Friis transmission equation [84] explains the transmitted energy from the UGS and received energy in the UAV have a relationship as

$$\begin{aligned} E_R &= \alpha \frac{G_T G_R \lambda^2}{(4\pi R)^2} \frac{1}{n} \left(\frac{1}{\beta_1^{l_1}} E_{T,1} + \cdots + \frac{1}{\beta_n^{l_n}} E_{T,n} \right), \\ \lambda &= \frac{c}{f}, \\ l_i &= 2\|T_i, h\| \sin\left(\frac{\theta_h}{2}\right), \quad i = 1, 2, \ldots, n \end{aligned} \tag{4.15}$$

where E_R is the energy received in the UAV, $E_{T,i}$ is the energy transmitted from the ith transmitter loop attached on the UGS, α is an overall efficiency degradation rate caused from other than transmitter and receiver loops, G_T and G_R are the transmitting and receiving gains, λ is the microwave wavelength in meters [85], c is the speed of light ($3E8\ m/s$), f is the frequency of the single-tone signal, R is the vertical distance between transmitter and receiver, n is the total number of transmitter or receiver loops, β is an energy transfer coefficient between the transmitter loop and the receiver loop dependent on the distance between two loops, l_i is a distance between T_i and R_i, T_i is the location of the ith transmitter loop, R_i is the location of the ith receiver loop, h is the center of rotation, and θ_h is the amount of rotation (Figure 4.7(a)). The h exists since the polygon shape consisted of $T_1, T_2, \ldots, T_n$ is designed to exactly match with $R_1, R_2, \ldots, R_n$ to maximize efficiency. Since transmitted energies from each transmitter loop are all same, I can simplify Equation 4.15 as

$$E_R = E_T \alpha \frac{G_T G_R \lambda^2}{(4\pi R)^2} \frac{1}{n} \sum_{i=1}^{n} \beta_i^{-l_i}. \tag{4.16}$$

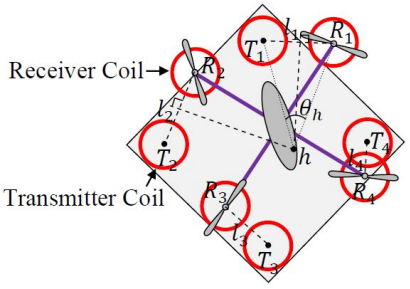
So, total system efficiency can be written as

$$\text{Efficiency} = \frac{E_R}{E_T} = \alpha \frac{G_T G_R}{(4\pi R)^2} \frac{c^2}{f^2} \frac{1}{n} \sum_{i=1}^{n} \beta_i^{-l_i}. \tag{4.17}$$

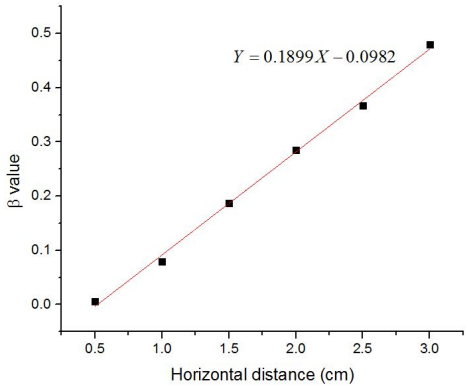
Since the inherent energy transfer efficiency between a transmitter and a receiver loops, $\frac{G_T G_R \lambda^2}{(4\pi R)^2}$, is 0.1332 and the inherent energy transfer efficiency of total system, $\alpha \frac{G_T G_R \lambda^2}{(4\pi R)^2}$, is 0.0785, a governing equation for the β can be found by considering one pair of loops ($n = 1$), applied voltage ($V_T = 17.89V$), received voltage (Figure 4.2(d)), and electrical power equation $P = \frac{V^2}{R}$ as

$$\beta_i = \left(\alpha \frac{G_T G_R}{(4\pi R)^2} \frac{c^2}{f^2} \frac{1}{n} \frac{V_T^2}{V_R^2} \right)^{\frac{1}{l_i}} = \left(0.0785 \frac{V_T^2}{V_R^2} \right)^{\frac{1}{l_i}}, \tag{4.18}$$

and β can be plotted with changing horizontal distance ($l_i = \{0cm, 0.5cm, \ldots, 3cm\}$) (Figure 4.7(b)).



(a) Misalignment between the UAV and UGV.



(b) Experiment results to calculate β.

Figure 4.7.Misalignment and corresponding β value change.

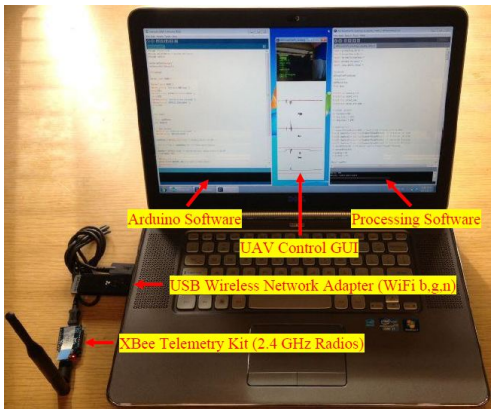
Since the governing equation of β can be expressed with a linear equation as $\beta_i = 0.1899 l_i - 0.0982$, Equation 4.17 can be rewritten for the application of four pairs of transmitter and receiver loops ($n = 4$) as

$$\begin{aligned}\text{Efficiency} &= \alpha\frac{G_T G_R}{(4\pi R)^2}\frac{c^2}{f^2}\frac{1}{n}\sum_{i=1}^{n}\left(0.3798\|T_i,h\|\sin\left(\frac{\theta_h}{2}\right)-0.0982\right)^{-2\|T_i,h\|\sin\left(\frac{\theta_h}{2}\right)}, \\ &= 0.0196\sum_{i=1}^{4}\left(0.3798\|T_i,h\|\sin\left(\frac{\theta_h}{2}\right)-0.0982\right)^{-2\|T_i,h\|\sin\left(\frac{\theta_h}{2}\right)}. \quad (4.19)\end{aligned}$$

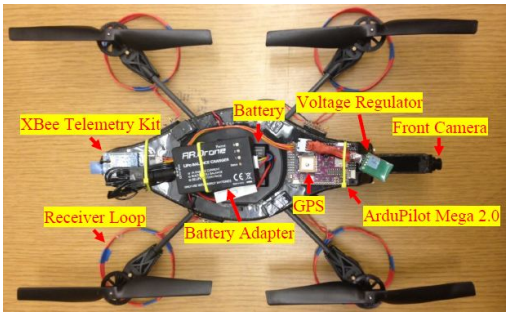
4.6 Experiments

An AR.Drone [86] designed by Parrot SA company (Figure A.1(b)) is used for experiments by integrating an ArduPilot Mega 2.0 (APM) microcontroller onto the AR.Drone. AR.Drone is controlled by Processing software on the UGS (Figure A.1(a)) using Wi-Fi (b,g,n) signals and APM can perform two-way communication by Arduino software on the UGS using $2.4GHz$ radio signals (Figure A.1(c)). AR.Drone System includes an AR.Drone, a MB1200 XL-MaxSonar-EZ0 ultrasonic range finder, a set of XBee telemetry kit, a AnyVolt Micro Universal DC-DC converter, and ArduPilot Mega 2.0 microcontroller which includes a MediaTek MT3329 GPS, 6-axis accelerometer/gyro, 3-axis magnetometer, and an ultrasonic altimeter. In addition, AR.Drone itself contains a 6-axis accelerometer/gyro, 3-axis magnetometer, an ultrasonic altimeter, and two cameras (640x480 pixels VGA). One pair of Accelerometers, gyros, and ultrasound sensors are integrated to achieve better data.

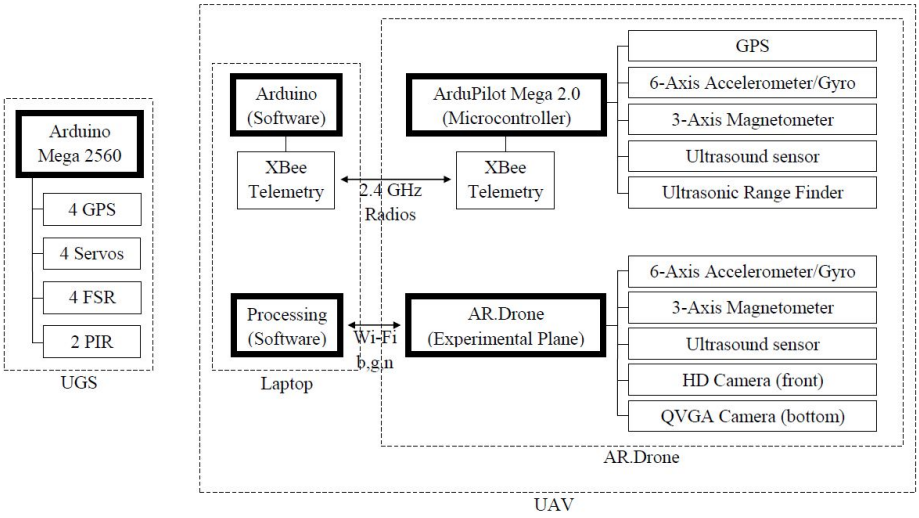
Autonomous recharging operation (Figure 4.9) is performed with an assumption that the AR.drone already arrives near the UGS. Once automatic landing algorithm starts, the AR.drone takes off (step 1) and begins surveillance to detect marks attached on the UGS by flying in spiral shapes and maintaining altitude (step 2). Two different type of marks and a total of five marks are attached on the UGS for the AR.drone to detect. The first design of one mark is attached at the center of the UGS and the second design of four marks are distributed around the center. Since using only one mark at the center does not guarantee for the AR.drone to detect it due to the video data loss with unreliable Wifi communication, four additional marks



(a) Ground Station System Set Up.



(b) AR.Drone System Set Up (20.7x20.3in diameter, 536g weight).

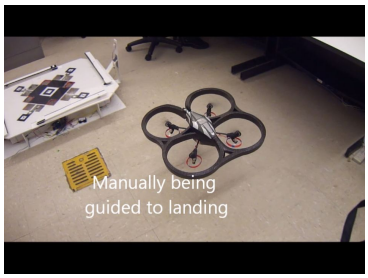


(c) Communication Set Up.

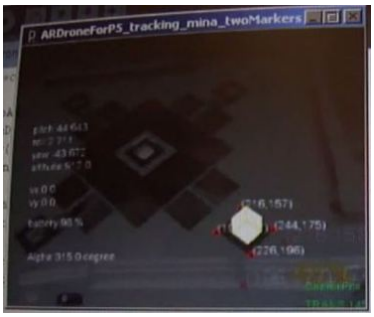
Figure 4.8.Experiment set up.

are introduced to increase the chance of detection. Once the AR.drone detects one or more boundary marks (step 3-4), the AR.drone tries to hover right above the boundary mark (step 5) and searches for a center mark. After the center mark is detected (step 6), the AR.drone uses the center coordinates of the image frame and the center coordinates of the center mark to let the AR.drone land on the UGS (step 7). Once landed, four bars on the UGS move in and position the AR.drone to the center (step 8) and the charging operation starts (step 9). Once charging is finished, the four bars retract (step 10) and the AR.drone leaves the UGS.

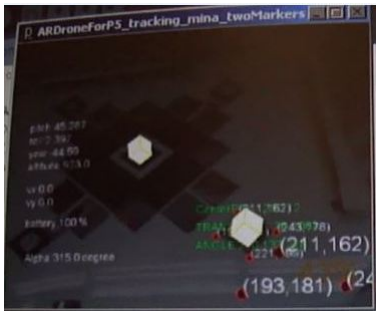
(a) Step 1.



(b) Step 2.



(c) Step 3.

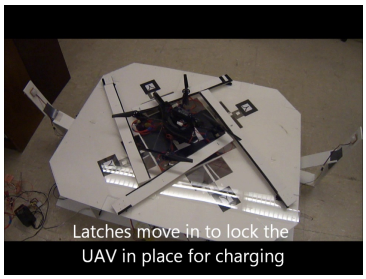


(d) Step 4.

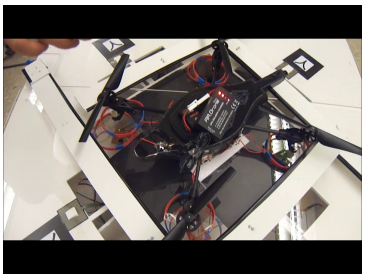
(e) Step 5.

(f) Step 6.

(g) Step 7.



(h) Step 8.

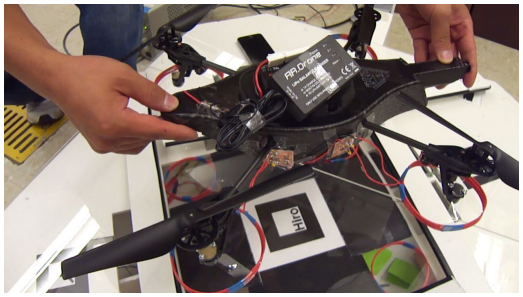


(i) Step 9.

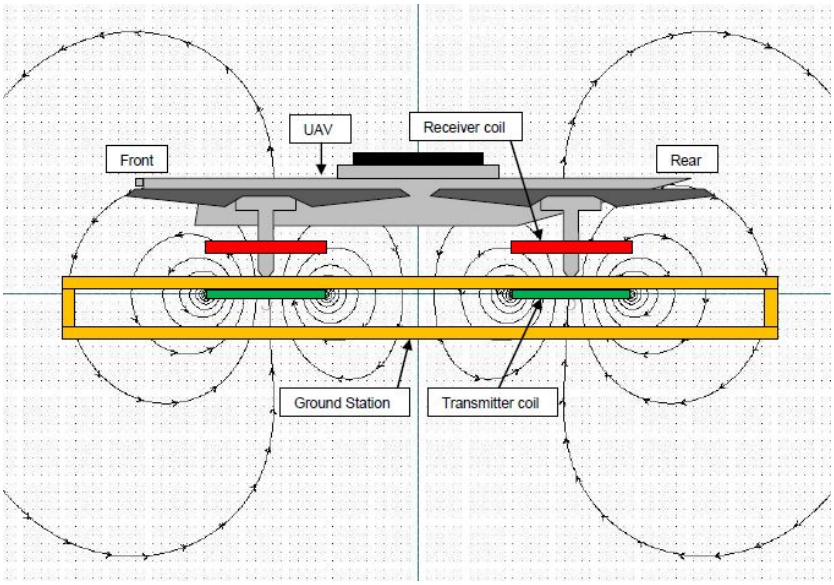
(j) Step 10.

Figure 4.9.Automatic charging operation using AR.drone.

(a) Charging works given valid distance.



(b) Charging fails given beyond distance.



(c) Simulated Magnetic field (side view).

Figure 4.10.Wireless charging demonstration using four transmitters and four receivers.

With input voltage $27.62V$ and current $0.957A$, charging starts (Figure 4.10(c)) when transmitters and receivers are close enough (red led lights on the AR.drone Charger turn on). However, when transmitters and receivers are placed beyond a certain distance, charging stops (Figure 4.10(b)). In addition, magnetic fields around each transmitter and receiver are simulated (Figure 4.10(d)). A video with detailed explanation is available at `http://www.youtube.com/watch?v=hzjDokU7YfA`.

4.7 Concluding Remarks

In this chapter, I designed a wireless charging station for AR.drone. Four transmitters send enough voltage and current to four receivers and a battery on AR.drone is successfully charged with the total system efficiency of 7.85%. In addition, an automatic landing algorithm for AR.drone also works fine using image detection algorithm to allow it to autonomously land. In the future, improved operation will be tested outside using GPS sensors. AR.drone will be placed some distance away from the UGS, then the UAV will be maneuvered to fly to the UGS and landed on it autonomously by considering the wind effect as well. A wind turbine and/or a solar panel will be integrated to operate outside without any wired power source. Also, to increase system power transfer efficiency, circuits driven with several MOSFETs will be integrated since the heat loss may linearly reduce as the number of MOSFET increases. Also, inductors with low resistance will be used to provide better power transfer from one side to the other.

CHAPTER 5: MAP UNCERTAINTY, ERROR PROPAGATION, AND ALGORITHMIC ROBUSTNESS

Highly automated mission planning for UAV brings almost proportional amount of error propagation and breaking this proportionality is a puzzling paradox. To prevent the domination of errors in developing fully autonomous UAV systems, accurate investigation of each error source should be performed. This chapter inquires into possible error sources in autonomous UAV systems for the simplest mission plan, trajectory generation of a single UAV. This chapter gives an idea to UAV system developers to evolve current remote piloted vehicle concept UAV to fully autonomous UAV with robustness.

5.1 Background and Motivation

As Unmanned Aerial Vehicle (UAV) technologies evolve, the mission planning tends to be automated with less operators [23]. This automated mission planning might give some feasible solutions to missions, but these feasible solutions may involve possible crashes without considering the robustness of automated processes since there are many unexpected situations such as high tension wires, trees, smoke plumes, suspended sand, birds/insects, other UAVs, cold, heat, icing, rain, fog, sleet, snow, hail, air pocket, wind, wind shear, and so on [87]. To verify robustness of the automated processes, I should investigate algorithmic errors that appear in every automated process in mission planning, environmental disturbances such as weather, system limitation, and system malfunctions, so that I can use the error propagation rate to avoid unexpected collisions.

Automated algorithmic processes used in this chapter were developed in a previous paper to provide safe flight trajectories for two types of UAVs (fixed-wing and hover-

able) preventing collisions into static objects [62]. The automated algorithm contains inner processes of automatic building detection [88], transformation into monochrome image, corner detection, Voronoi Diagram [89–95], Dijkstra [96, 97], multiple Traveling Salesman Problem (mTSP) [98, 99], Genetic Algorithm (GA) [100–103], and so on. Unfortunately, the algorithms will be refused to be used in real environment without detailed investigation since the algorithm contains some amount of inherent errors and accumulation of these errors might result unsafe flight trajectories. So, it is important to investigate the robustness of the proposed algorithm thoroughly.

Concerning the environmental disturbances (mostly wind disturbances), S. Jackson proposed a spatial sliding mode controller and a receding sliding mode controller to compensate for the 2D wind effect [104]. According to the flight experiment results, the kinodynamic controller resulted about 20% less mean sensor path error with almost same standard deviation than the sliding mode controller. Also, T. Shima presented the transformation of inputs from no-wind to wind scenario with associated transient response [23]. Including the previous two research works, most researchers deal with the wind effect by including constant or sinusoidal wind velocity, v_{wind}, in the vehicle dynamic equation.

Most UAVs possess propulsion systems such as propellers or engines which require fuels in the form of gas, electricity, hydrogen, hybrid, and so on. Due to the limitation of fuel carriage, appropriate consideration on fuel state should be investigated to operate UAV as a part of health monitoring [75]. J. Smith took into account the fuel rate for the fuzzy priority for helping the requester [87]. Also, for swarming operations using multiple UAVs, updating overall system health information is vital to avoid possible collisions. Specifically, persistent surveillance using group health management is accomplished using MIT RAVEN testbed [105] by considering uncertain fuel usage dynamics [106–108]. Due to the significance of the subject, robustness analysis based on the fuel state of the proposed algorithm of this chapter is separated from here.

In my work, I investigate the error propagation in each step in the hierarchy of mission planning, and integrate those errors to get an overall error. Then, with the analyzed overall errors, I present a method to choose safe buffer zone size of static objects. The combinatorial explosion of error propagation is possible when the maximum errors from each step are integrated, but it is unusual case. In this chapter, I focus on the error propagation of the normal case and my approach to decomposing the problem and error propagation can give specific guarantees for specific accuracy of the maps or measurements if there are normal errors propagation. To accomplish the goal, I introduce the overall hierarchy of mission planning in section 5.2, investigate error propagations in each step of the hierarchy in section 5.3, comparison between pure and error dominated mission planning hierarchy in section 5.4 , and finally present concluding remarks and future works in section 5.5.

5.2 Hierarchy of Mission Planning

UAV path planning using GA based mTSP was solved in a previous work where scalability of autonomous UAV was attempted but not robustness [62]. Robustness of the automated UAV mission planning on a given geometry can be proved by analyzing disjointed error propagations at each step since each algorithm runs independently, in series, and in one direction (Figure 5.1).

Overall, there are five major disturbances in mission planning; photograph error, corner detection error, combinatorial error, vehicle dynamic error, and environmental disturbances. Due to these errors and disturbances, the trajectory generated at the end of the mission planning hierarchy does not guarantee the UAVs' crash avoidance. However, if the amount of error propagation in the mission hierarchy can be estimated, I can minimize UAV collision rate by choosing appropriate size of the buffer zone around buildings (Figure 5.2(d)) and also reduce misdetection rate of the building and corner detection processes by assigning appropriate number of human operators.

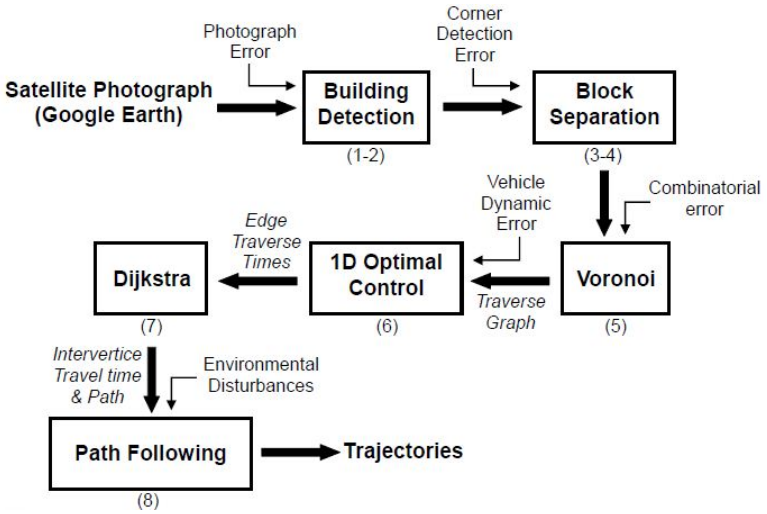
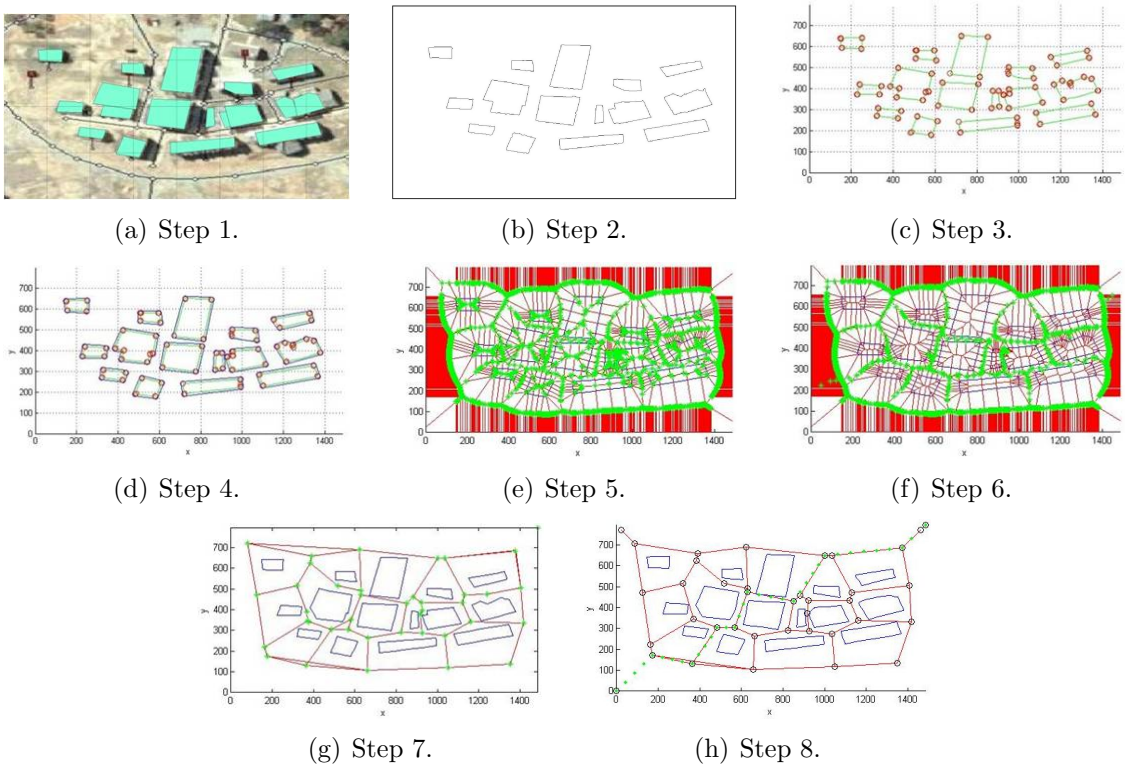


Figure 5.1.Hierarchy of mission planning.



(a) Step 1. (b) Step 2. (c) Step 3.

(d) Step 4. (e) Step 5. (f) Step 6.

(g) Step 7. (h) Step 8.

Figure 5.2.Processes to generate a trajectory of a single UAV.

5.3 Disjunction of Algorithmic Error

5.3.1 Step 1: Photograph Error

The hierarchy of mission planning (Figure 5.1) starts with a given satellite photograph extracted from Google Earth (Figure 5.2(a)). Most of pure Google Earth

photographs are taken by WorldView I and II satellites designed by DigitalGlobe and have the resolution up to $50cm$ [109], but Google Earth itself which uses 436 control points has much worse positional accuracy of $39.7m$ RMSE ($0.4m <$ error $< 171.7m$) [110, 111]. Some research works propose a method to use georeferencing to increase the data point accuracy to a positional accuracy of 5-6m [110]. The georeferencing method uses the recorded GPS coordinate data at each corner of the map snapshot using Garmin eTrex summit HC handheld GPS unit to match with the composite satellite map. The positional accuracy of 5-6m is mostly due to the device used for the experiments, Garmin eTrex summit HC handheld GPS unit, and it also means that I can increase the accuracy by using a GPS device with higher accuracy. For instance, if I use a Sokkia GSR2700 ISX for the georeferencing, I will be able to achieve an accuracy upto $1.5m$ CEP [112, 113] and I use this accuracy for the rest of this chapter with an assumption that the area where I want to perform surveillance is well known beforehand.

The mission hierarchy (Figure 5.1) uses a constant buffer size, c, to avoid possible vehicle crashes, so I can set c as $1.5m$ as the buffer zone size. However, there are still 50% of chance to have data collected from the outside of a circle with $1.5m$ radius, so I need to increase the buffer zone size to be $c = 1.5N_{ph}$ where N_{ph} is a safety factor for the photograph error. For instance, if I use $N_{ph} = 2$, then my buffer zone size will be $3m$. To be conservative, I would be better to choose as large N_{ph} as possible, but the introduction of the large safety factor also brings some detours in the original trajectories requiring more fuel consumption ($N_{ph} \propto N_f$, where N_f is the total amount of fuel consumption (unit: kg)). Therefore, it will be necessary to have appropriate decision making to maximize the safety and minimize the fuel consumption.

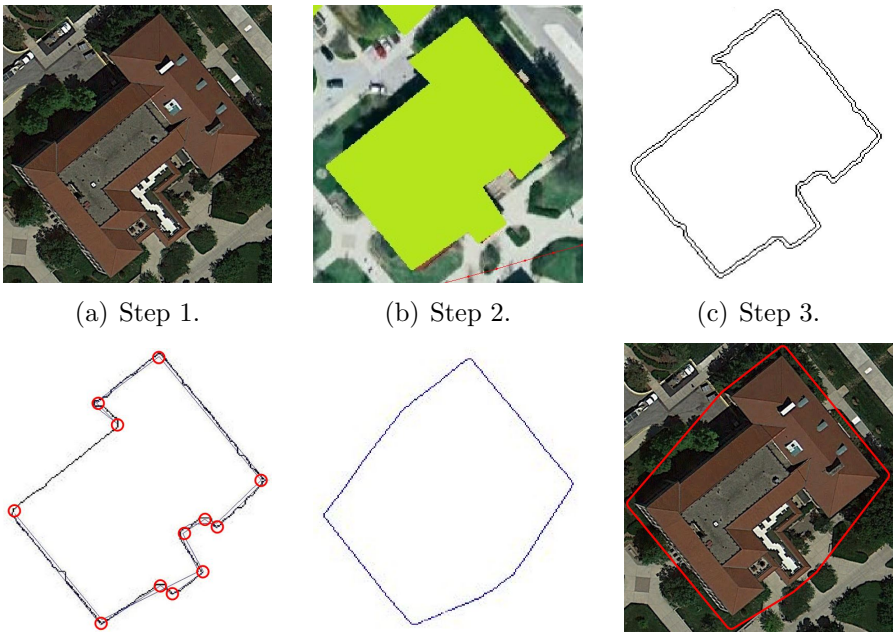
5.3.2 Step 2: Building Detection Error

The latest automated building extraction algorithm using IKONOS images [114] has 83.2% building detection rates thanks to the IKONOS satellite which has $1m$ or slightly better spatial resolution. To increase the building detection rate, I can run the algorithm n times on the same surveillance region so that I can decrease the amount of undetected building as $(1 - 0.832)^n = 0.168^n$. For instance, for a region with 15 buildings (Figure 5.2(a)), the automated building extraction algorithm might miss to detect in the amount of $15 \cdot 0.168 = 2.52$ buildings, but the number of missed buildings might be decreased as $15 \cdot 0.168^2 = 0.42$ buildings if I run the algorithm two times and so on.

5.3.3 Step 3: Corner Detection Error

Corner detection algorithm was greatly enhanced up to 98.14% by using adaptive threshold and dynamic Region of Support (ROS) to take into account both global and local curvature of corners [115]. Since the coordinates of the extracted buildings are transformed into MATLAB® using the corner detection algorithm (Figure 5.1(c)), accurate detection of obstacle corners is as important as the detection of obstacles to avoid possible collisions. With detected corners, I draw a polygon which wraps the poorly detected edges (Figure 5.3(e)). The corner detection algorithm is not efficient to detect buildings with curvatures and it results about 96.16% correctness (that is, 3.84% error) on wrapping the buildings (Figure 5.4) in respect of,

$$\text{Corner detection rate} = \frac{\text{Area of the correctly detected buildings}}{\text{Area of the total buildings}}. \tag{5.1}$$



(a) Step 1. (b) Step 2. (c) Step 3.

(d) Step 4. (e) Step 5. (f) Step 6.

Figure 5.3.Processes to detect buildings with $2m$ buffer zone size.

Figure 5.4.Wrapping the detected buildings with $2m$ buffer zone size.

5.3.4 Step 5: Combinatorial Error

Three sources of the combinatorial error in the Voronoi diagram algorithm are proposed [116]; distance error (due to the incorrect depth comparison of a pixel);

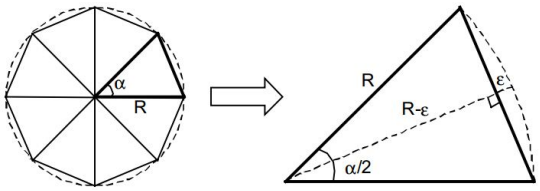
resolution error (due to the coarse discrete sampling); Z-buffer precision error (due to the precision limitations of bits in graphic systems). With an assumption that there is no Z-buffer precision error, the error bound can be expressed as

$$dist(P, A) \leq dist(P, B) + 2\varepsilon, \tag{5.2}$$

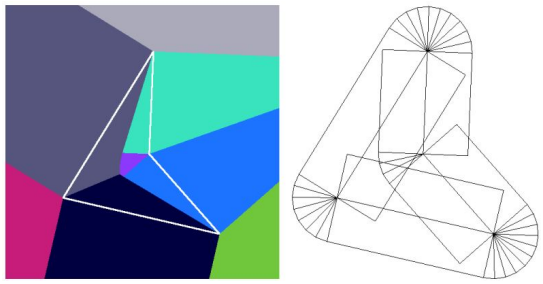
where $dist(P, A)$ is the distance from the center of pixel P to the site A and ε is the maximum distance error. By using the equation in the paper [116],

$$\cos\left(\frac{\alpha}{2}\right) = \frac{R - \epsilon}{R}, \tag{5.3}$$

where α is the acute angle of the isosceles triangles (1024×1024 has 85 triangles) and R is the radius of the cone (max distance between site and sample point) as shown in Figure 5.5, I can calculate ϵ as



(a) "A single triangle of the meshed point distance function cone. α is the angle we wish to maximize, R is the radius of the cone (max dist between site and sample pt), and ε is the max error."



(b) "The per-feature Voronoi diagram of a quadrilateral (left). The corresponding distance mesh (right)."

Figure 5.5.Descriptions about how to calculate α and R [116].

$$\begin{aligned} \epsilon &= R\left(1 - \cos\left(\frac{1}{2}\frac{2\pi}{85}\right)\right), \\ &= R(6.83 \times 10^{-4}). \end{aligned} \tag{5.4}$$

So, I can get to the conclusion that ϵ is small enough to be ignored even if R is several hundred meters. This brings the fact that the combinatorial error in Step 5 which is mainly caused by the Voronoi diagram algorithm can be ignored.

5.3.5 Step 6: Vehicle Dynamic Error

Vehicle dynamic error is really the propagation of sensing error and so I study how much the sensing error exists in my vehicle dynamic model. The dynamic model of the UAV used in this chapter [117] considers two more factors compare to the general UAV dynamic models such as position tracking time constant (τ_x) and velocity tracking time constant (τ_v) which mostly come from the data communication delay. Introduction of the τ_x and τ_v results more realistic vehicle simulation in the sense that there are normally some amount of communication delay in real life UAV deployment. The model is proposed as,

$$\begin{aligned} \mathbf{x}_c(k+1) &= \mathbf{x}_c(k) + T\mathbf{v}_c(k), \\ \mathbf{v}_c(k+1) &= -\frac{T}{\tau_x\tau_v}\mathbf{x}_c(k) + \left(1 - \frac{T}{\tau_v}\right)\mathbf{v}_c(k) + \frac{T}{\tau_x\tau_v}\mathbf{x}_c^{ref}(k), \end{aligned} \tag{5.5}$$

where, x_c is the position vector of the UAV ($\in R^3$), v_c is the velocity vector of the UAV ($\in R^3$), T is the sampling time, and $\mathbf{x}_c^{ref}$ is the tracking reference points ($\in R^3$). Since there are vehicle dynamic errors in position and velocity terms, Equation 5.5 becomes

$$\begin{aligned} \mathbf{x}_c(k+1) &= \mathbf{x}_c(k) + \delta\mathbf{x}_c(k) + T\left[\mathbf{v}_c(k) + \delta\mathbf{v}_c(k)\right], \\ \mathbf{v}_c(k+1) &= -\frac{T}{\tau_x\tau_v}\left[\mathbf{x}_c(k) + \delta\mathbf{x}_c(k)\right] + \left(1 - \frac{T}{\tau_v}\right)\left[\mathbf{v}_c(k) + \delta\mathbf{v}_c(k)\right] + \frac{T}{\tau_x\tau_v}\mathbf{x}_c^{ref}(k), \end{aligned} \tag{5.6}$$

and if I gather the error terms, Equation 5.6 becomes

$$\begin{aligned}
\mathbf{x}_c(k+1) &= \mathbf{x}_c(k) + T\mathbf{v}_c(k) + \left[\delta\mathbf{x}_c(k) + T\delta\mathbf{v}_c(k)\right], \\
\mathbf{v}_c(k+1) &= -\frac{T}{\tau_x\tau_v}\mathbf{x}_c(k) + \left(1 - \frac{T}{\tau_v}\right)\mathbf{v}_c(k) + \frac{T}{\tau_x\tau_v}\mathbf{x}_c^{ref}(k) + \\
&\quad \left[-\frac{T}{\tau_x\tau_v}\delta\mathbf{x}_c(k) + \left(1 - \frac{T}{\tau_v}\right)\delta\mathbf{v}_c(k)\right].
\end{aligned} \tag{5.7}$$

Since I am using $\tau_x = 0.25s$, $\tau_v = 0.5s$, and $T = 0.01s$, the amount of errors in position and velocity become

$$\begin{aligned}
e_p &= \delta\mathbf{x}_c(k) + 0.01\delta\mathbf{v}_c(k), \\
e_v &= 0.08\delta\mathbf{x}_c(k) + 0.98\delta\mathbf{v}_c(k),
\end{aligned} \tag{5.8}$$

where, e_p represents the position error and e_v represents the velocity error. The nonlinear UAV dynamic model used in this chapter describes the motion of vehicle well, but errors still appear mostly due to inherent sensor errors.

I investigate errors from MediaTek MT3329 GPS sensor [118] used for acquiring position data and MPXV7002DP Airspeed sensor [119] used for acquiring velocity data. The MediaTek MT3329 GPS sensor has horizontal position accuracy as $3m$ CEP. With the safety factor for the position error, N_{po}, I assume the GPS sensor has $3N_{po}$ amount of position error. The MPXV7002DP Airspeed sensor can measure pressures from -75 to $75kPa$ with the maximum error of 6.25%. With a simple calculation, I can find the measurable acceleration range as $\dot{v} = 75 \cdot 1000 \cdot A/K$ where v is velocity in m/s, $1kPa$ is $1000N/m^2$, A is area of the pitot tube in m^2, and K is weight of the UAV in kg. With the measured area of the pitot tube [119] ($A = \pi(0.00231/2)^2 = 4.191 \cdot 10^{-6}m^2$) and weight of RC UAV Drone [120] ($K = 2.7kg$), I can calculate the $\dot{v}$ as about $0.12m/s^2$ and so the measurable acceleration range turns out to be from -0.12 to $0.12m/s^2$ with a possible error of $0.015m/s^2$. Since I am more interested in the position sensor error, I can derive e_p using Equation 5.8 as,

$$e_p = 3N_{po} + 0.01 \cdot 0.015 = 3N_{po} + 1.5 \cdot 10^{-4} \approx 3N_{po}. \quad (5.9)$$

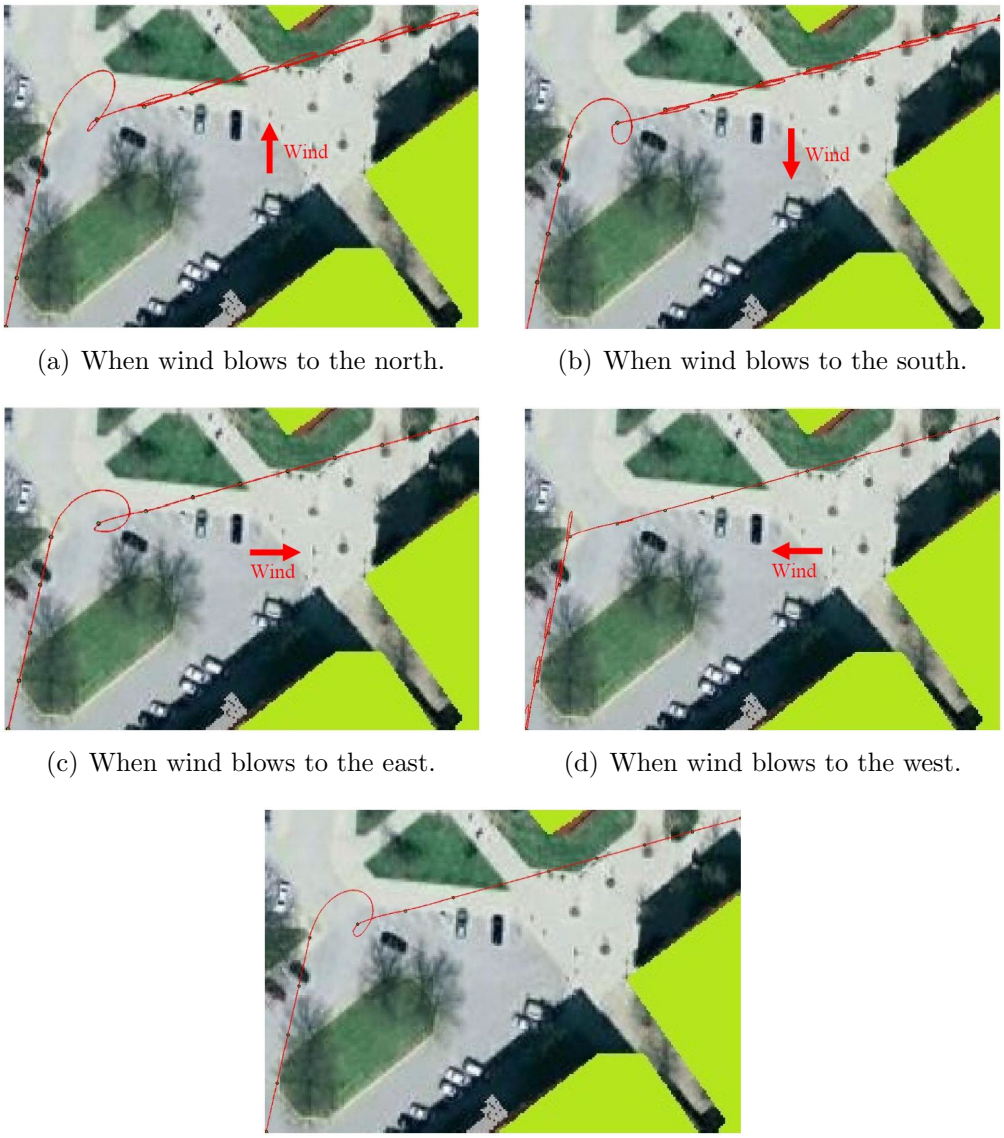
5.3.6 Step 8: Environmental Disturbances

The most prominent environmental disturbance on the UAV flight is the wind effect. I can add the wind effect to Equation 5.5 as,

$$\mathbf{x}_c(k+1) = \mathbf{x}_c(k) + T\mathbf{v}_c(k) + T^2\frac{1}{2}a_w(k), \quad (5.10)$$

where a_w represents the amount of UAV acceleration caused by the wind ($\in R^3$) and I assume the magnitude of a_w is $0.1m/s^2$ in this chapter. Wind disturbance causes the UAV to fly slightly off from each node (Figure 5.6(a-d)) which does not happen when there is no wind (Figure 5.6(e)). For instance in Figure 5.6(a), the UAV can follow nodes well when it flies to the north direction since the wind blows to the north. However, when the UAV flies to the east, the UAV keeps circulating to pass through the predefined trajectory nodes due to the wind disturbance. Not only the north direction, but also the south, east, and west directions of wind disturbances cause similar phenomena. Depends on the direction of the wind, the UAV trajectories result slight shifts to the direction of the wind as expected.

Overall trajectory error of each wind disturbance direction is calculated by finding the area between two lines; the predefined trajectory (green nodes) and the actual UAV trajectory (red lines). The error turns out to be; north direction results $3.3 \pm 0.23m$; south direction results $2.8 \pm 0.19m$; east direction results $2.2 \pm 0.16m$; west direction results $2.1 \pm 0.18m$. From this, I can expect that there will be maximum trajectory error of $3.3 \pm 0.23m$ due to the wind disturbance.



(a) When wind blows to the north.

(b) When wind blows to the south.

(c) When wind blows to the east.

(d) When wind blows to the west.

(e) When wind does not exist.

Figure 5.6.UAV trajectories when wind causes the UAV to accelerate with magnitude of $0.1m/s^2$.

5.3.7 Overall Algorithmic Error

All of the individual error can be listed as,

1. Step 1 (photograph error): $1.5N_{ph}$ RMSE photograph position error (unit: m),

2. Step 2 (building detection error): $100 \cdot 0.168^n\%$ automated building detection error,

3. Step 3 (corner detection error): 3.84% corner detection error,

4. Step 5 (combinatorial error): No error,

5. Step 6 (vehicle dynamic error): $3N_{po}$ position error (unit: m),

6. Step 8 (environmental disturbances): $3.3 \pm 0.23m$.

If I choose $N_{ph} = 3$ and $N_{po} = 2$, I have a total error as $e_{t1} \approx 1.5 \cdot 3 + 3 \cdot 2 + 3.3 = 13.8m$, where e_{t1} represents the total error of the mission planning hierarchy (Figure 5.1). However, not only the overall algorithmic error, but I also need to incorporate UAV system constraints, such as the minimum forward velocity v_{min}, maximum forward velocity v_{max}, maximum acceleration a_{max}, and minimum turn radius $r_{min} = \frac{v_{min}^2}{a_{max}}$, to achieve much safer operation beyond the algorithmic error. First of all, I give a schematic of the derivation of r_{min} in Figure 5.7. Here, two edges intersect at an angle α, b_1 is the distance in which the UAV decelerates to its minimum velocity, and b_2 is the distance to the intersection at which it begins to turn inside circle c_2 to avoid collision. Having v_{min} and v_{max} is for the application of 1D optimal control in the mission planning hierarchy (step 6 in Figure 5.1).

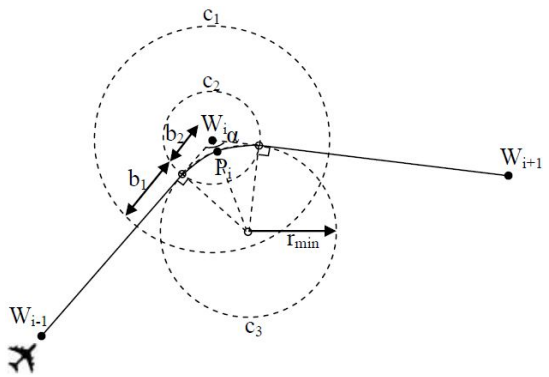


Figure 5.7.Minimum radius r_{min} which allows a fixed-wing UAV to change its direction without any collision.

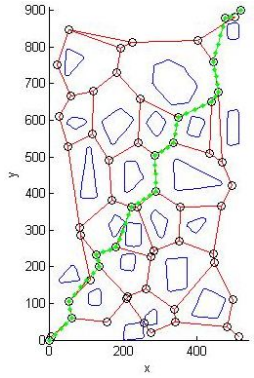
The time for deceleration is $t_d = \frac{v_{max}-v_{min}}{a_{max}}$. Hence, $b_1 = v_{max}t_d - \frac{1}{2}a_{max}t_d^2 = \frac{v_{max}^2-v_{min}^2}{2a_{max}}$. Tangent lines to the circle c_3 result in the distance b_2 which is calculated as $b_2 = r_{min}\tan\left(\frac{\pi}{2}-\frac{\alpha}{2}\right) = \frac{v_{min}^2}{a_{max}}\tan\left(\frac{\pi}{2}-\frac{\alpha}{2}\right)$. The distance, b_2, depends upon α which is determined by the way points, w_{i-1}, w_i, and w_{i+1}. So long as the arc of motion in Figure 5.7 is within the width of the path between obstacles, i.e. road width $w_R \geq \overline{W_iP_i} = \sqrt{b_2^2 + r_{min}^2} - r_{min} = r_{min}\left(cosec\frac{\alpha}{2} - 1\right)$, the turn is traversable. For the fixed-wing UAV system properties listed in Section 5.4, I can calculate $e_{t2} = b_1 + b_2 = 3.75 + 0.29 = 4.04m$ with $\alpha = 120^\circ$ for the fixed-wing vehicle. Thus the buffer is clearly dependent upon vehicle dynamics and the sharpest turn I expect the vehicle to execute.

Therefore, I should use the final buffer zone size as the sum of e_{t1} and e_{t2}, that is, $17.84m$ for the safest flights. In addition, from step 2 and step 3, I can expect the sum of the automated building and corner detection errors in percentage as $e_s = 1 - (1 - 100 \cdot 0.168^2)(1 - 0.0384) = 0.0655$ by choosing $n = 2$. This brings the conclusion that the UAV might have collision due to undetected buildings or corners of buildings with the 7% chance. To avoid the collision, I might need human resource to eliminate or decrease the misdetection rate.

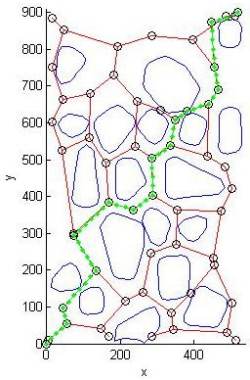
5.4 Algorithmic Robustness Analysis

All simulations are done with a fixed-wing vehicle flying from the starting location, [0,0], to the goal location, [472,808], which has following system properties; $v_{min} = 0.5m/s$, $v_{max} = 2m/s$, $v_{initial} = [0,0,0]m/s$, $a_{max} = 0.5m/s$, $Altitude_{min} = 3m$, $Altitude_{max} = 50m$, $Altitude_{normal} = 30m$, time step of SIMULINK® as $0.01s$, $\tau_x = 0.25s$, and $\tau_v = 0.5s$. All simulations are done using a desktop with Intel(R) Core(TM) 2 Duo CPU 3.00 GHz Processor, 64-bit Operating System, and 4.00 GB RAM. In Figure 5.8, red lines represent candidate trajectories and green lines are selected trajectories for the UAV.

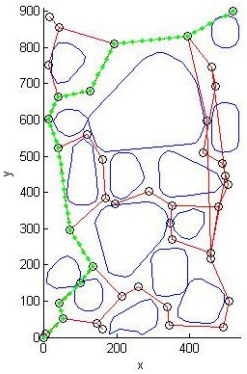
When the buffer size is increased, some of overlapped obstacles are combined together (Figure 5.8(b,c)). In contrast to the smaller buffer size, the larger buffer size eliminates some trajectory candidate lines and this brings different trajectory outputs compare to the smaller buffer size. The trajectory output of 17.84m (Figure 5.8(c)) guarantees that UAV will be in safe regardless of photograph error (step 1), combinatorial error (step 5), vehicle dynamic error (step 6), and environmental disturbances (step 8).



(a) UAV trajectories only with e_{t2} (4.04m) buffer size.



(b) UAV trajectories only with e_{t1} (13.8m) buffer size.



(c) UAV trajectories with sum of e_{t1} and e_{t2} (17.84m) buffer size.

(d) UAV trajectories only with e_{t2} (4.04m) buffer size (Google Earth).

(e) UAV trajectories only with e_{t1} (13.8m) buffer size (Google Earth).

(f) UAV trajectories with sum of e_{t1} and e_{t2} (17.84m) buffer size (Google Earth).

Figure 5.8.Robustness verification by changing the buffer size of buildings (Squirrel park at Purdue University (lat: 40.422108°, lon: -86.932187°)).

5.5 Concluding Remarks

Error propagation of the proposed automated UAV trajectory generation algorithm is studied. In the mission planning hierarchy, I generally have photograph error, building detection error, corner detection error, combinatorial error, vehicle dynamic error, and environmental disturbances. Sum of this overall errors turns out to be $e_{t1} = 13.8m$ with $e_s = 7\%$ UAV collision rate coming from the misdetection of buildings or corners of buildings. Also, I introduce additional buffer size of $e_{t2} = 4.04m$ by incorporating UAV system constraints and so I conclude the final buffer size as $e_{t1} + e_{t2} = 17.84m$ which guarantees safe flights of the fixed-wing UAV even if multiple errors are appeared in the automated mission planning algorithm. To decrease the UAV collision rate coming from the building and corner detection errors, I need to hire human resources to identify undetected buildings or corners of buildings during the automated UAV trajectory generation algorithm processes. Identification of the least number of human resources will be studied in the future.

PART 3: SCALABILITY

CHAPTER 6: MINIMUM TIME INTEGRATED SURVEILLANCE AND RECONNAISSANCE (ISR)

I developed a toolkit to enable large scale autonomy with guarantees of safe performance for UAVs by trading plenteous memory with limited online processing in the course of automating all of the aspects of mission planning. I combat the complexity and uncertainty of unstructured environments through *offline pre-computation*, building in robustness to various uncertainties, and the use of feedback in real-time operation. I first convert maps into traversability graphs using both map features and vehicle dynamic constraints through Voronoi type algorithms. Second, I obtain the geometric paths and the traversal times for all of the shortest paths in the region for various vehicles using 1-D time optimal control. In both the generation of traversability graphs and in 1-D optimal control, I supply safety margins to account for map and sensor inaccuracy. This converts standard surveillance and search into a multiple Traveling Salesman Problem (mTSP) for the vehicles over the graphs for minimum time performance. I develop region division to convert the mTSP for multiple vehicles into TSPs for individual vehicles depending upon their initial locations–the algorithms used are NRD (Normal Region Division), GKVRD (Gustafson-Kessel Fuzzy Clustering with Voronoi Region Division), and KVRD (K-means Clustering with Voronoi Region Division). I solve these TSPs offline using GAs (Genetic Algorithms) subject to collision constraints with neighboring regions. I illustrate with some simulations of real-time operation, including mission changes and vehicle failure.

6.1 Background and Motivation

Operational autonomy for UAVs is the decision making capacity of the UAV to attain operational objectives. As the Level of Operational Autonomy (LOA) increases,

the uncertainties faced by the vehicle increase combinatorially as does the complexity of its decision making as seen in Table 6.1 [23].

Table 6.1Level of Operational Autonomy.

LOA	Value	Function
High	10	Fully Autonomous
Autonomy	9	Battle Space Swarm Cognizance
	8	Battle Space Cognizance
	7	Battle Space Knowledge
	6	Real-Time Multi-Vehicle Cooperation
	5	Real-Time Multi-Vehicle Coordination
	4	Fault-Event Adaptive Vehicle
	3	Robust Response to Real-Time Faults/Events
	2	Changeable Mission
Low	1	Execute Preplanned Mission
Autonomy	0	Remotely Piloted Vehicle

The scope and complexity of Unmanned Aerial Vehicle (UAV) missions have made autonomous real-time operation in unstructured environments elusive [25, 121]. Hence, the dependence on multiple human operators per UAV is perhaps the largest cost component of operating them [26]. Operator attention and effort are subject to fatigue and error [27], operators are subject to severe stresses, especially in combat operations, something that can be avoided if tasks are suitably automated [28]. There are many efforts under way to increase the autonomy of UAVs [23, 29, 30]. The difficulties arise from the propagation of several uncertainties, many dynamic, into the performance of these systems–map errors, modeling errors, sensor errors, errors of actuation, wind gusts, EM and acoustic interference, enemy maneuvers, and cyberattacks [31]. Arbitrary combinations of these uncertainties make it impossible to predict the performance of most battlefield systems as they are operated today. Several research efforts into autonomous systems involve heuristic solutions to combinatorial optimization problems that do not supply guarantees in real-time [122], even if solved in real time [102], where Genetic Algorithms (GAs) were solved on Field Programmable Gate Arrays (FPGAs). Although [98, 100] give a solution for even

mTSP type problems to minimize mission completion time using modified GA, the solution achieved is not the best for surveillance in the sense that some routes are overlapped. However, if the mission is the attacking of targets, the avoidance of route overlaps is not an important constraint compared to the timing, visitation angle, and collision avoidance as studied in [101], where GAs are again used for optimization.

In this work, I convert a large class of ISR (Integrated Surveillance and Reconnaissance) problems for passive targets in mapped regions to mTSPs (multiple Traveling Salesman Problem) [99] which are solved offline to provide several alternative solutions stored offline for online use. I construct an end-to-end framework integrating existing algorithms where available and developing my own where necessary, to solve this class of problems. My framework incorporates vehicle dynamics and mission constraints so the GA solving the mTSP only searches over feasible solutions. In the terms of Table 6.1, I aim at an LOA of 5.

I formulate the problem class in Section 6.2, mapping it to a discrete combinatorial optimization problem in Sections 6.3 and 6.4, and decomposition of mTSPs for a team of UAVs to TSPs (Traveling Salesman Problem) for m UAVs via partitioning of space in Section 6.5. Scalability in the sense of computational time with the number of map features and UAVs is discussed in Section 6.6, and I conclude in Section 6.7.

6.2 Problem Formulation

I assume here that the entire group of vehicles has only a single objective of minimizing the expected time to complete a mission. In mathematical programming language:

$$\begin{aligned} \min \quad & E\left[t_{mission}\right] \\ \text{s.t.} \quad & \left(\begin{array}{l} \text{Vehicle dynamic constraints} \\ \text{Environmental constraints} \\ \text{Sensor and actuator constraints} \\ \text{Modeling and map errors} \\ \text{Mission requirements} \end{array} \right), \end{aligned} \tag{6.1}$$

where $E\left[t_{mission}\right]$ is the expected time required for the mission. All of the constraints are stochastic in nature. Hence I can only minimize the expected value of the time for a mission. I illustrate my framework with coordinated surveillance. The number of vehicles or individual service demands are all incorporated into the constraints which are dynamic. I consider the class of problems in which mission requirements are expressed as spatio temporal constraints on vehicle states by operators through suitable user interfaces. Thus operator load in this case is proportional to the frequency of requests. Scenarios such as 'eyes on target,' persistent surveillance, co-ordinated search, and target tracking with handoffs are included within this class. However, the vastly more difficult problem of interpreting sensory data is not included. In a complex situation, interpretation of data can cause operator load to increase arbitrarily. Hence for a first cut, I assume the interpretation of sensory data is simple enough so as to be automated. Examples include landmark recognition and recognition of well characterized objects. My mission planning algorithm in Figure 6.1 has seven hierarchical steps–each corresponding to the entry of specific uncertainties to which robustness needs to be built in to permit autonomous operation;

1. First, the Region Growing Algorithm (RGA) is used to generate a monochrome map from a two dimensional satellite image.

2. Second, the map is analyzed to detect blocks and number them using the vertices of blocks.

3. Third, construction of an area or volume Voronoi diagram [90] converts the blocks into a traversability graph. I convert this to traversability graphs customized to individual vehicles based on abilities to round corners and turns in tight spaces.

4. Fourth, 1D optimal control between graph vertices converts all graph edges into minimum traverse times for specific vehicles. Basically, UAVs are managed to accelerate at a vertex, maintain a constant velocity, and decelerate before reaching the next vertex.

5. Fifth, I use Dijkstra's algorithm to generate all pairs shortest (minimum time) paths between all graph vertices using the traverse times between neighboring vertices from Step 4.

6. Sixth, I solve mTSPs of visits to all the vertices using the pairwise minimum times obtained in Step 5.

7. Seventh, the vehicles perform real time path following of the offline paths according to user requests for search, surveillance and tracking.

The mTSPs are solved through decomposition into m TSPs, one for each of the m vehicles. While I do this decomposition using a Voronoi diagram based on the Euclidean metric for the mapped region, other more natural metrics, such as a Manhattan distance, are likely to yield better results. There have been many researches on path planing using Voronoi diagrams such as [36, 123–126] which used point obstacles to generate paths and [91, 93, 127] which used a set of polygons as obstacles to generate paths. The common technique is that UAV trajectories resulting from Voronoi diagrams are used as initial inputs for further smoothing processes. This smoothing process differentiates the contribution of each paper. To the best of my knowledge, my hierarchy of mission planning is unique compared to the previous papers due to the fully automated process from satellite photographs to feasible trajectories by incorporating vehicle, map, and mission constraints where necessary.

Real time ISR problems are solved through use of stored solutions in combination with feedback. By feedback, I mean reactive maneuvers such as speeding up or slowing down on a path, and getting back on course after wind gusts. The feasibility and robustness of my stored solutions require both accurate map information and sufficiently accurate geo-location. Sensitivities to map error and navigation error depend strongly on the local geometry of the map. Hence robustness of my solutions in a given area can be estimated a priori using error propagation in the algorithms depicted in Figure 6.1. Error propagation has been quantified previously for map building [128] and Voronoi algorithms [89]. For 1D optimal control, error estimates can be obtained through use of H_∞ system norms [129]. No additional errors are introduced in my use of Dijkstra's algorithm or GA to solve TSPs as these steps only involve finding minimum sums of traverse times obtained in Step 5.

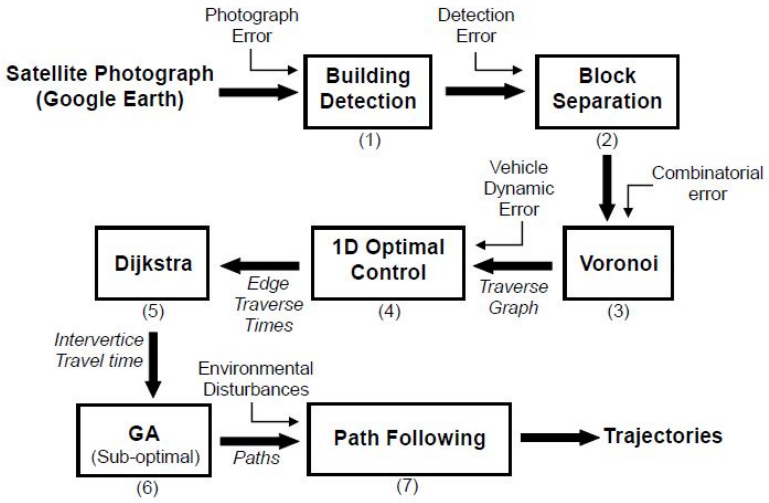


Figure 6.1.Hierarchy of mission planning.

The typical complaint against soft computing methods does not hold in my case as I am using them to solve problems offline to produce feasible rather than optimal solutions. Moreover, I only use GA to search among feasible solutions rather than searching over all solutions. Second, these methods are relatively easier to setup ad hoc in unpredictable environments with unpredictable mission requirements or constraints. As more of the problem structure becomes known, exploitation through

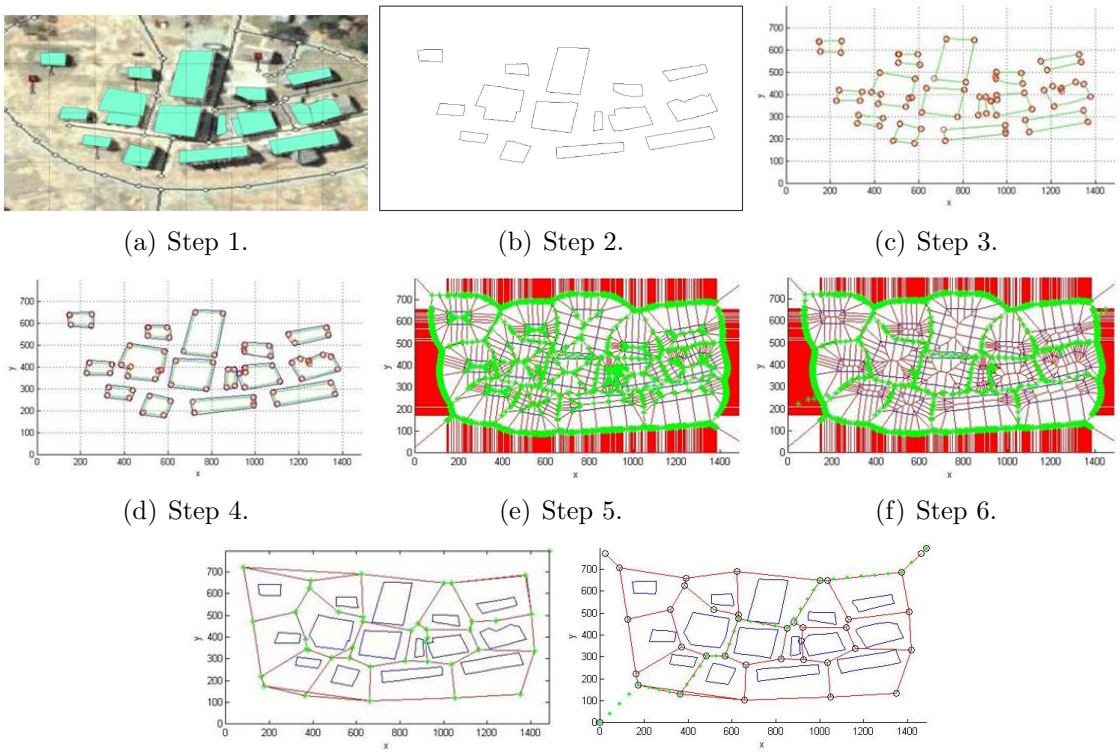
hard computing/optimization methods may become possible even under the time constraints. Given the availability of several feasible offline solutions, they can be chosen randomly in real time to avoid system compromise to cyberattacks. However this will necessitate longer mission times. The formal statement of my mTSP is as follows.

$$
\begin{aligned}
\min \quad & E\left[t_{mission} = \sum_{k=1}^{m}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}x_{ijk}\right) + w_k + o_k\right)\right] \qquad (6.2)\\
\text{s.t.} \quad & \sum_{j=2}^{n} x_{ijk} = 1 \quad \text{for any } k,\\
& \sum_{j=2}^{n} x_{jik} = 1 \quad \text{for any } k,\\
& \sum_{i\neq j} x_{ijk} = 1 \quad \text{for any } k,\\
& \sum_{i=j} x_{ijk} = 0 \quad \text{for any } k,\\
& c_{ijk} = G_t(i,j), \quad \text{for} \quad i \neq j,\\
& w_k, o_k \geq 0,
\end{aligned}
$$

where c_{ijk} is the time taken by kth UAV between ith and jth node, x_{ijk} is a constant between { 0,1 }, n is the number of vertices, m is the number of UAVs, G_t is the time array among nodes (n × n), w_k is the time increase due to wind, o_k is the time increase due to unexpected obstacles. The strategy encoding schema for the GA that solves the TSP for individual vehicles is described in Appendix B.

6.3 Discretization of Space

Buildings and structures are discerned in satellite or aerial images either manually or automatically by using Automatic Building Detection algorithm [88, 114] followed by RGA. Here, I note that [88] assumes that the building structures have convex



(a) Step 1. (b) Step 2. (c) Step 3.

(d) Step 4. (e) Step 5. (f) Step 6.

(g) Step 7. (h) Step 8.

Figure 6.2.Processes to generate a path.

rooftop sections. In case the rooftops are not convex, the algorithm obtains convex hulls of the detected corners.

This chapter begins with the assumption that buildings are detected using the method described in [88] as shown in Figure 6.2(a). Then, RGA is applied to extract the edges and corners of the buildings as shown in Figure 6.2(b). The Block Separation Algorithm (BSA) (Appendix B) is then used to number blocks and save all the corners of buildings as shown in Figure 6.2(c). Since a UAV might collide into buildings due to errors in maps, sensors and actuation, wind gusts and other disturbances, the blocks are extended with a buffer zone around them with my Polygon Extension Algorithm (PEA) detailed here. A simple calculation of the buffer for accurate maps for a given vehicle is given later in the section. In Figure 6.2(d), blue lines drawn around each building are the buffer zone and the method of drawing this is explained

in Figure 6.3. Here, $P^c_{i,j}$ is the jth point at the ith corner C_i where $i \in \{1, ..., n_p\}$, n_p is the total number of vertices of the given polygon, and $j \in \{1, 2, 3\}$. Three points $(P^c_{i,1}, P^c_{i,2}, P^c_{i,3})$ are chosen per each corner by drawing a circle with radius of buffer zone size. Two end points per each corner $(P^c_{i,1}, P^c_{i,3})$ satisfy $\overrightarrow{P^c_{i,1}, C_i} \perp \overrightarrow{C_i, C_{i-1}}$ and $\overrightarrow{P^c_{i,3}, C_i} \perp \overrightarrow{C_i, C_{i+1}}$ where $C_0 = C_{n_p}$ and $C_{n_p+1} = C_1$. Among three points, $P^c_{i,1}$, $P^c_{i,2}$, and $P^c_{i,3}$, the middle point $P^c_{i,2}$ is chosen to satisfy $\angle P^c_{i,1} C_i P^c_{i,2} = \frac{1}{2} \angle P^c_{i,1} C_i P^c_{i,3}$, so that buffer zones are drawn big enough to surround blocks and also decrease the amount of computation time with less data points to compare. Three points, $P^e_{i,1}$, $P^e_{i,2}$, and $P^e_{i,3}$, per each edge are chosen to satisfy,

$$\frac{1}{4} \mid P^c_{i,3}, P^c_{i+1,1} \mid = \mid P^c_{i,3}, P^e_{i,1} \mid = ... = \mid P^e_{i,3}, P^c_{i+1,1} \mid . \tag{6.3}$$

These points at the corners and edges of polygons are used as an input for construction of the Voronoi diagrams as shown in Figure 6.2(e). The crossing points of the selected red lines are the nodes used to generate paths for UAVs. In order for UAVs to avoid obstacles, vertices located inside the blocks are removed in two dimensional traversability mapping, but those are counted in three dimensional traversability mapping by placing nodes over the buildings which will lead UAVs to fly over the buildings, but not into the spaces inside buildings. The remaining vertices are represented as green '$*$' as shown in Figure 6.2(f). In Figure 6.4, I eliminate closely spaced vertices on line l_k produced by Voronoi methods (arising from 'roughness' of the map), i.e., vertices that are not intersections of three edges. An intersection point of three edges is N_k found with vector analysis where it exists and included as a node for the traversability graph. This sometimes results in some straight edges L_k intersecting the buffered obstacles. I add additional vertices to avoid straight line paths intersecting obstacles. The vertex $n_{k,p}$ added as N_{new} is the point on the curved edge l_k that minimizes the distance $d(P^c_{i,j+1}, n_{k,p})$ to $P^c_{i,j+1}$ on the buffered obstacle as

$$N_{\text{new}} = n_{k,\arg\min_p \left[d(P^c_{i,j+1}, n_{k,p}) \right]}, \tag{6.4}$$

where $p \in \{0, 1, ..., q\}$ and q is the total number of nodes on line l_k. The resulting traversability graph is shown in Figure 6.2(g). An example shortest path is shown in Figure 6.2(h).

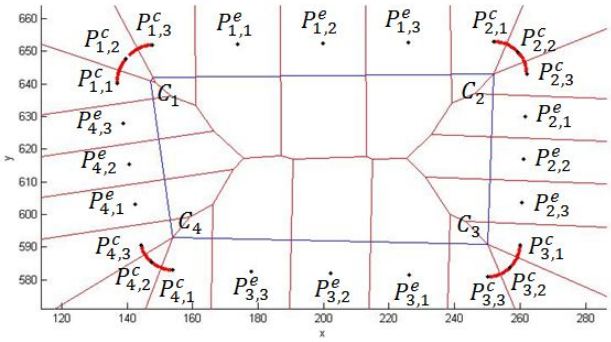


Figure 6.3.Choosing 3 points from each corner and edge.

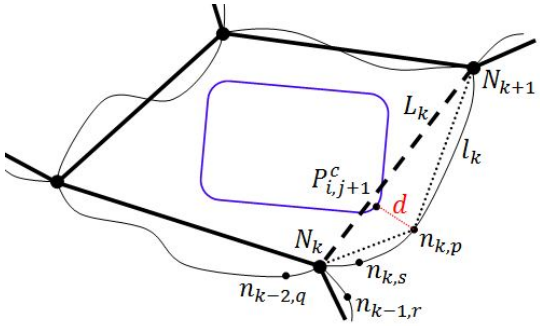


Figure 6.4.Extra vertices for collision free straight line paths.

I modify the traversibility graph for fixed-wing vehicles to incorporate their constraints of minimum forward velocity v_{min}, maximum acceleration a_{max}, and minimum turn radius $r_{min} = \frac{v_{min}^2}{a_{max}}$. I give a schematic of the derivation of r_{min} in Figure 6.5. Here, two edges intersect at an angle α, b_1 is the distance in which the UAV decelerates to its minimum velocity, and b_2 is the distance to the intersection at which it begins to turn inside circle c_2, to avoid collision.

The time for deceleration is $t_d = \frac{v_{max} - v_{min}}{a_{max}}$. Hence, $b_1 = v_{max}t_d - \frac{1}{2}a_{max}t_d^2 = \frac{v_{max}^2 - v_{min}^2}{2a_{max}}$. Tangent lines to the circle c_3 result in the distance b_2 which is calculated as $b_2 = r_{min} \tan\left(\frac{\pi}{2} - \frac{\alpha}{2}\right) = \frac{v_{min}^2}{a_{max}} \tan\left(\frac{\pi}{2} - \frac{\alpha}{2}\right)$. The distance, b_2, depends upon α which

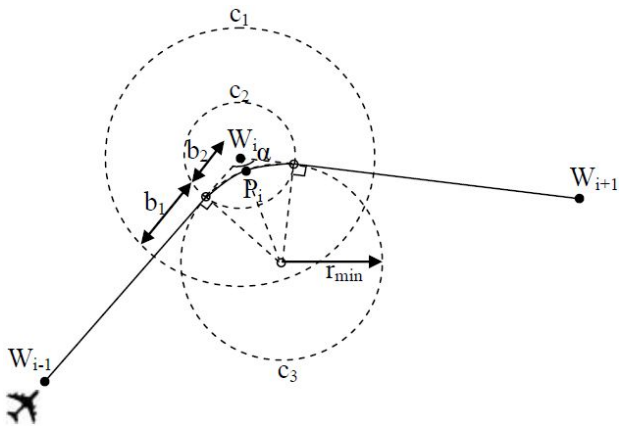
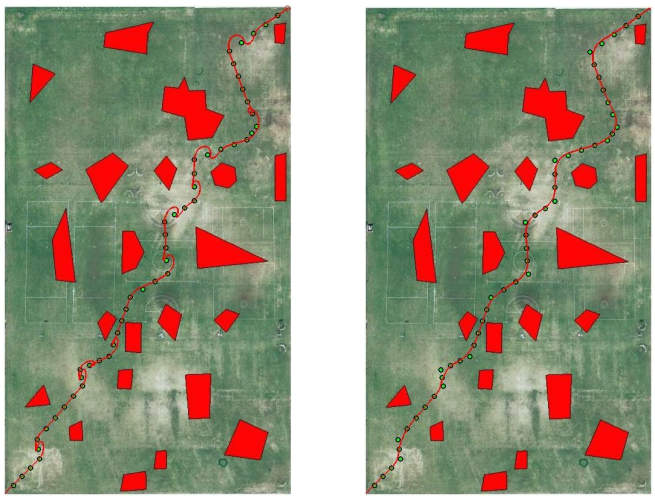


Figure 6.5.Minimum radius r_{min} which allows a fixed-wing UAV to change its direction without any collision.

is determined by the way points, w_{i-1}, w_i, and w_{i+1}. So long as the arc of motion in Figure 6.6 is within the width of the path between obstacles, i.e. road width $w_R \geq \overline{W_i P_i} = \sqrt{b_2^2 + r_{min}^2} - r_{min} = r_{min}\left(cosec\frac{\alpha}{2} - 1\right)$, the turn is traversable. Use of this calculation ensures collision free trajectories as shown in Figure 6.6(b) for the trajectory of a fixed-wing UAV. Although I am using v_{min} to be conservative to avoid crashes in Figure 6.6(b), higher velocities can be also used with larger c_1, c_2, and c_3 circles to reduce flight time.

An appropriate buffer zone value can be decided for the PEA. If the size of the buffer zone is too large or blocks are located closer than a threshold $l_{overlap}$, PEA merges blocks into one block. I calculate $l_{overlap} = 2\left(b_1 + b_2\right) = 20m$ with $\alpha = 120°$ for the fixed-wing vehicle of Appendix C. Thus the buffer is dependent upon vehicle dynamics and the sharpest turn I expect the vehicle to execute.



(a) UAV trace without using the r_{min} technique.

(b) UAV trace using the r_{min} technique.

Figure 6.6.The efficiency of the r_{min} technique.

6.4 Discretization of Time

Prior work in trajectory generation uses either variational methods [130] or receding horizon control methods [45, 131] that approximates their solutions. Variational methods use either optimality conditions and boundary conditions or adjustable dynamic variables to find a global optimal solution, and typically require heavy computations. My approach, in contrast, provides an approximate solution that is very conservative. My work involves speed control along prespecified paths for minimum time traverse while ensuring satisfaction of all vehicle turning constraints. I use UAV dynamics to find the shortest traverse times of the vehicle on all shortest paths on the graph, assuming minimum velocity traverse at all turns. UAV models for trajectory tracking from prior work [117] used for calculations in this chapter are accurate approximations to the dynamics of attitude stabilized vehicles. This is because global or semi-global closed loop stabilization transforms UAV dynamics into target dynamics of point mass models with definite tracking time constants for position, τ_x, and velocity, τ_v. Path following aircraft dynamics which is dominated by the slowest or

dominant poles is far more accurate than simplistic point mass or constant velocity models. The model is summarized below:

$$\begin{aligned} \mathbf{x}_c(k+1) &= \mathbf{x}_c(k) + T\mathbf{v}_c(k), \\ \mathbf{v}_c(k+1) &= -\frac{T}{\tau_x\tau_v}\mathbf{x}_c(k) + \left(1 - \frac{T}{\tau_v}\right)\mathbf{v}_c(k) + \frac{T}{\tau_x\tau_v}\mathbf{x}_c^{ref}(k), \end{aligned} \tag{6.5}$$

where T is the sampling time, x_c is the position of UAV, v_c is the velocity of UAV, and x_c^{ref} is the tracking set point for UAV. Next, traversability graphs drawn in the Euclidean space are converted into graphs in ECEF coordinates [132]. Once the Euclidean distance obtained from relative sizes in images is scaled up to the real world, I can use the distances in ECEF coordinates and UAV velocities to find the approximate traverse time.

The typical profile of velocity over a graph edge and the time intervals where the vehicle accelerates, moves at constant velocity, or decelerates are depicted in Figure 6.7 where p_{acc} is the position where a UAV accelerates, p_{con} is the position where a UAV velocity is constant, p_{dec} is the position where a UAV decelerates, t_{acc} is the time during which a UAV accelerates, t_{con} is the time during which a UAV velocity is constant, and t_{dec} is the time during which a UAV decelerates. Here, I discretize time intervals using the sample time T and calculate appropriate values of t_1, t_3, d_1, and d_3 via numerical integration and obtain the time of traverse of a vehicle from node $W_{i,k}$ to node $W_{j,k}$ as $t_{ij} = t_1 + t_2 + t_3$. The traverse time corresponding to all the shortest paths on the traversability graph G_d is thus calculated to produce its time equivalent graph G_t.

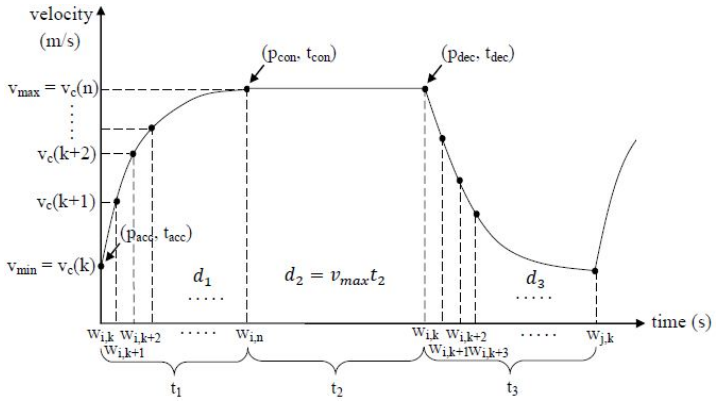


Figure 6.7.Discretized graph with step time, T.

6.5 Coordinated ISR

Surveillance of an area consists in 'keeping an eye' on it, e.g., UAVs visit all nodes and edges in a region within a time interval with specified frequency. I solve here the problem of visiting all nodes on the graph once with m UAVs. This boils down to an mTSP. To ease solution, I break up the mTSP into m TSPs for the m UAVs through two methods:

1. Uniform Region Division (URD) where the rectangular area is arbitrarily divided into m rectangular regions of the same area (Figure 6.8(a)).

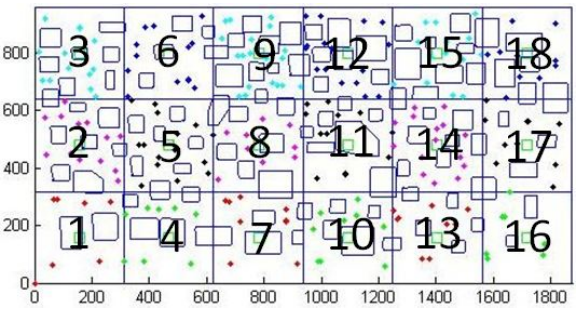
2. K-means Voronoi Region Division (KVRD) [133] where K-means clustering is used to generate centroids of graph nodes for which a Voronoi partition is constructed to divide the region into m pieces (Figure 6.9(a)).

In this chapter, both methods divide the region containing 103 buildings into $m = 18$ regions and each region is assigned to one of the UAVs. I show the simulated trajectories in Figure 6.8 and 6.9 and the flight times of fixed-wing UAVs in seconds in Table 6.2. In Figure 6.8(a) and 6.9(a), numbers represent UAVs assigned to each region. The maximum flight time of KVRD is $196s$ longer than the one of URD. So, KVRD takes a longer time to take UAVs back at the end of the mission planning. If high fuel efficiency is required for a mission, I should use KVRD since it has smaller

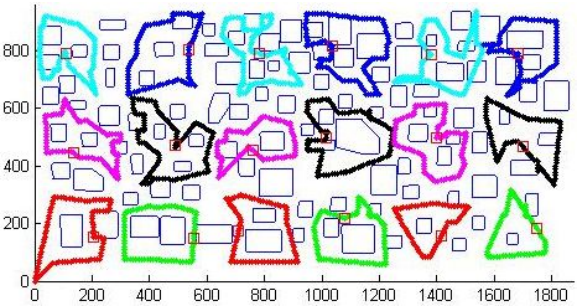
Table 6.2Estimated traveling time of Figure 6.8(c) & 6.9(c).

Order of UAVs	URD (s)	KVRD (s)
1^{st}	1000	805
2^{nd}	967	1509
3^{rd}	1017	934
4^{th}	762	678
5^{th}	1254	976
6^{th}	994	1047
7^{th}	837	1122
8^{th}	864	940
9^{th}	1313	734
10^{th}	845	1022
11^{th}	1156	699
12^{th}	1267	998
13^{th}	700	1012
14^{th}	985	800
15^{th}	1299	964
16^{th}	730	1049
17^{th}	1046	1275
18^{th}	1068	1220
Max	1313	1509
Min	700	678
Mean	1005.8	988

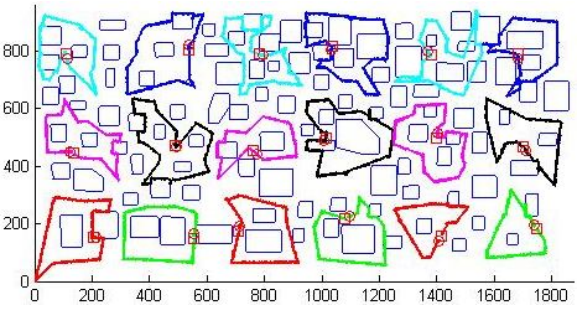
mean value than URD on this map. This may not apply to other maps, but with my procedure of offline solutions, the better method can be found a priori. In Figure 6.8(c,d) and 6.9(c,d), red square marks represent the launching sites of UAVs and red circular marks represent the arrival sites of UAVs. My calculations are feasible for arbitrary numbers of both buildings and UAVs subject only to constraints of offline computational resources. I describe these requirements in the next section.



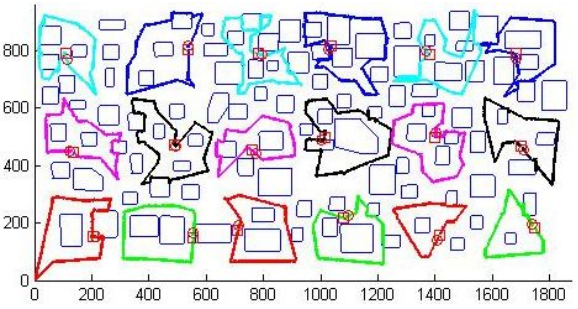
(a) Region division.



(b) Preplanned trajectories.

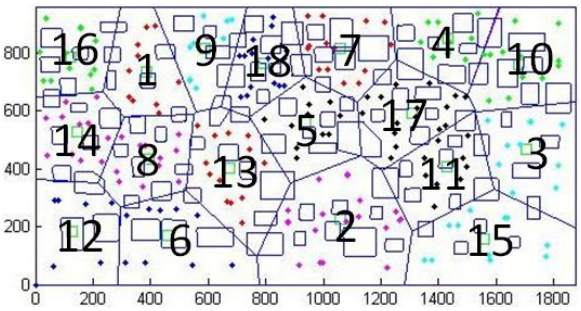


(c) Simulated flight trajectories with fixed-wing UAVs.

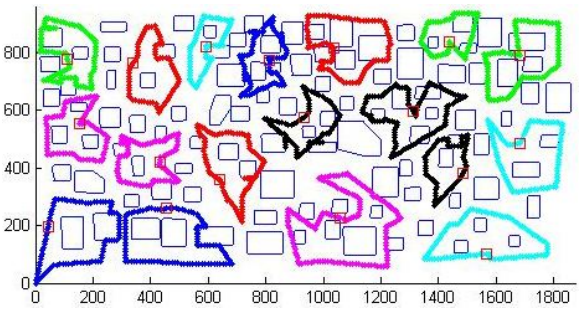


(d) Simulated flight trajectories with hover capable UAVs.

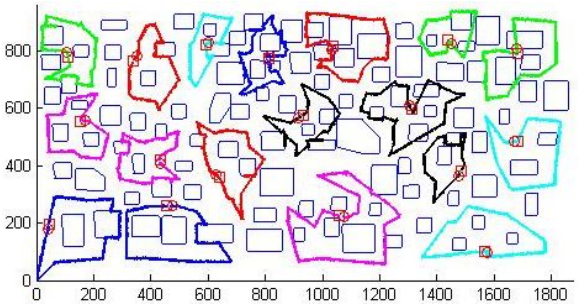
Figure 6.8.Application of URD (103 buildings and 18 UAVs).



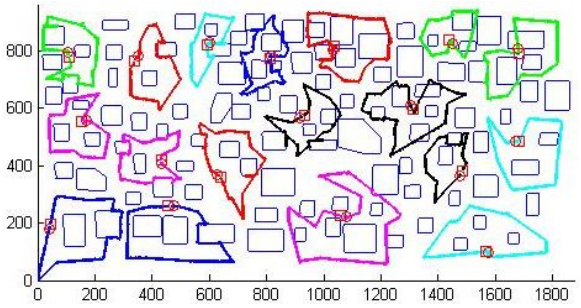
(a) Region division.



(b) Preplanned trajectories.



(c) Simulated flight trajectories with fixed-wing UAVs.



(d) Simulated flight trajectories with hover capable UAVs.

Figure 6.9.Application of KVRD (103 buildings and 18 UAVs).

6.6 Scalability Analysis

6.6.1 Offline Computational Complexity

The offline computational complexity of my algorithmic framework in Figure 6.1 is the total number of operations in steps 1-7 of my hierarchy:

$$\begin{aligned}
\text{Building Detection} &= O(|B||C|), \\
\text{Block Separation} &= O(|B||C|), \\
\text{Voronoi [95]} &= O(|n|log|n|), \\
\text{1D Optimal Control} &= O(|E||m_v|), \\
\text{Dijkstra [134]} &= O(|E| + |n|log|n|), \\
\text{GA} &= O(|n||P||N||m|), \\
&= O(|n|^2|P|^2), \\
\text{Path Following} &= O(|n|),
\end{aligned} \tag{6.6}$$

where B is the number of the buildings, C is the number of polygonal sides of the buildings, n is the total number of vertices, E is the total number of edges, m_v is the number of vehicle types (here, $m_v = 1$), P is the population size in the GA, N is the number of iterations in the GA, and m is the number of UAVs. The overall computational complexity of my framework C_F is therefore the sum of individual complexities:

$$\begin{aligned}
C_F &= O(|B||C|) + O(|B||C|) + O(|n|log|n|) + O(|E|) \\
&\quad + O(|E| + |n|log|n|) + O(|n|^2|P|^2) + O(|n|), \\
&\approx O(|n| + |n| + 2|n|log|n| + 2|E| + |n|^2|P|^2 + |n|), \\
&\approx O(2|n|log|n| + |n|^2|P|^2 + 5|n|),
\end{aligned} \tag{6.7}$$

using the fact that $O(|n|) \approx O(|B||C|) \approx O(|E|)$ for my maps and the number of iterations for the GA in Appendix A.

6.6.2 Online Computational Complexity

Once mission planning is completed, UAVs do not need additional computation time during online mission planning other than the path following step ($O(|n|)$) except in some emergency situations:

1. Unmapped obstacles.
2. Losses of UAVs (including through sensor failures).
3. Wind gusts.

In the first scenario, if a number of obstacles appears on the trajectories at different locations, I generate new candidate trajectories around the new obstacles using Voronoi diagrams. Then, 1D Optimal Control on the new candidate trajectories is applied and finally the shortest paths detouring the new obstacles are chosen. Hence, the additional computation time is

$$\begin{aligned} C_1 &= O(|n_{\text{part}}|log|n_{\text{part}}| + |E_{\text{part}}| + |E_{\text{part}}| + |n_{\text{part}}|log|n_{\text{part}}| + |n_{\text{part}}|), \quad (6.8) \\ &= O(2|n_{\text{part}}|log|n_{\text{part}}| + 3|n_{\text{part}}|), \end{aligned}$$

where n_{part} is the number of vertices around the new obstacles and E_{part} is the number of edges around the new obstacles.

In the second scenario, if some UAVs are lost during a mission, there are generally two options to complete the mission:

1. Let the remaining UAVs complete their missions first and then finish the paths of the lost UAVs using stored plans for smaller map segments.

2. Use stored plans with smaller number of UAVs out of the remaining UAVs to search the remaining areas.

The first option requires

$$C_2 \quad = \quad O(|E_{\text{part}}| + |n_{\text{part}}|log|n_{\text{part}}| + |n_{\text{part}}|), \tag{6.9}$$

amount of additional computation. The second option does not require any additional computation outside of waypoint following since stored plans are used.

In the third scenario, if the trajectories of some UAVs are disturbed by strong wind gusts, the interrupted UAVs can continue missions by choosing one of two options:

1. Fly to the previous waypoints.

2. Continue to fly to the next waypoints if the sensing task is not compromised.

The first option does not introduce any additional computation since the amount of trajectory disturbance to wind gusts is usually small. Sometimes, some UAVs are displaced much farther, so it is likely that they are lost due to collisions with obstacles.

Thus, except for poorly mapped regions, my algorithmic framework does not require very much of real time computation outside of waypoint following in individual vehicles.

6.7 Concluding Remarks

I have developed an end-to-end algorithmic framework in an attempt to attain scalable autonomous operations. In order to do so, I have used existing algorithms where available, and developed my own where necessary. These include my discretization of space and time, and partitioning of mTSPs into m TSPs. My framework permits solution of search, surveillance, tracking and handoff problems. I have illustrated my methods with a surveillance problem. Out framework may not give feasibility

for specific user requests because of its inherent conservation–buffers around buildings, assuming slower vehicle speeds, and the decomposition of mTSPs into m TSPs. The most prominent questions that now lie open to rigorous solutions within my framework are:

1. Systematic calculation of algorithmic sensitivities for the propagation of uncertainty through the algorithms, and selection of buffers and thresholds to state and prove formal robustness properties in a given mapped region.

2. Minimum fuel or maximum endurance operation of UAV teams (as opposed to the minimum time solutions that I have presented).

3. Taking advantage of vehicle and sensor heterogeneity in dividing up missions, and taking account of vehicle and sensor orientations along trajectories.

4. Efficient partitioning of perturbed mTSPs and feasible solutions to the resulting TSPs in real-time. This is needed to address real time requirement and resource changes.

While I can make the Voronoi partitioning of mTSPs more efficient in practice with better metrics, proofs of improved efficiency will be difficult. Many problems that require intensive operator effort can potentially be automated through use of more sensing. Hence, over time, I can automate most of the solution of specific mission planning problems, i.e., those without significant strategic uncertainty for the time being. In my surveillance problem, I exploit the predictability of the target and its environment to the extent possible. I only expect to obtain feasible solutions most of the time, and not all of the time as mentioned above. There is the possibility of operator overload and operation failure on occasion, as mentioned above. This does not conflict with the No Free Lunch Theorem [135]. Another way of looking at these problems is that the construction of optimal solutions is computationally intensive while being sensitive to modeling assumptions. On the other hand, solution procedures finding sub-optimal solutions with robustness may not find feasible solutions

even if they exist. Hence, I can hope to minimize the attentional effort of operators most of the time, but not eliminate it.

CHAPTER 7: MAXIMUM FUEL EFFICIENCY

Multiple Traveling Salesman Problem (mTSP) with n number of nodes to be visited by m identical UAVs can be solved by transforming into m Traveling Salesman Problems (TSP) and then solved with improved Genetic Algorithm (GA). Within my framework of spatio-temporal discretization and offline calculation, this is done with the objective of maximal fuel efficiency or minimal fuel consumption. While m TSPs correspond to m regions are running, each UAV under this algorithm can exchange nodes to equalize total traveling distances of each UAV. Exchanged nodes are prioritized in the shortest distance between two adjacent regions among m regions. Region which has longer total traveling distance gives a node located most closely to the neighboring region which has shorter total traveling distance. This give and take process will continue until differences among total traveling distances of each region become less than a certain threshold. Equalized traveling distances of each UAV keeps all UAVs in the air for the about the same time, and it increases operational efficiency. Extensions to heterogeneous UAV types are discussed toward the end of this work.

7.1 Background and Motivation

Multiple Traveling Salesman Problem (mTSP) can be solved using various methods. [99, 136–138] However, none of previous works considered fuel efficiency of Unmanned Aerial Vehicles (UAVs) as another constraint for optimization methods. In real life, military uses mostly homogeneous UAVs for surveillance and so those UAVs can carry approximately equal amount of fuels. In order to maximize surveillance coverage, UAVs should be in operation as long as possible until those fuels are consumed enough. If total distance of each trajectory resulted from mTSP method is

biased to specific trajectories, then UAVs flying these trajectories will consume fuels quickly and return to the base station. Rather, having equalized trajectories guarantees better fuel efficiency by equally distributing and consuming fuels. In addition, fuel efficiency can be maximized by letting UAVs fly with the minimum velocity since high velocity introduces bigger value of drag force [139] as, $F_d = \frac{1}{2}\rho v^2 c_d A$, where ρ is the mass density of the fluid, v is the speed of the object relative to the fluid, c_d is the drag coefficient, and A is the reference area. The drag coefficient and the velocity is proportional each other.

Fuel consumption management for a single UAV by using the glider type UAV (described as soaring flights through linear wind gradients) was proposed and formulated [140]. It was only applied to a single UAV, but it would be a good approach to extend multiple glider type UAVs to minimize fuel consumption. Fuel consumption management for the group of UAVs is formulated for the persistent surveillance coverage by considering the vehicle failures and degradations [106], but this work focuses only on the fuel consumption level of UAVs by excluding the consideration on the surveillance coverage. In contrast, surveillance coverage is dealt with a cost function to achieve better fuel efficiency by accommodating the threat avoidance and path length [141], but still this work lacks the application of the proposed method to the multiple UAVs surveillance problem. In some works, genetic algorithm (GA) based Traveling Salesman Problem (TSP) is used for the path planning of the multiple vehicles, but the optimization formulation setup in those works did not consider the method of finding the optimum number of vehicles to minimize the fuel consumption to do surveillance mission and also the fuel carriage limit per each vehicle [100]. The work done by Naval Research Laboratory [87] was in succeed to integrate various variables including fuel, obstacles, sample points to visit, and mission complete time, but it lacks the mentions on the intelligent methods of UAV distributions to each cite.

To compensate the listed problems, I develop the hierarchy of mission planning (Figure 7.1) which results the optimized number of vehicles to obtain the minimum

fuel consumption and the minimum mission complete time when I perform surveillance missions using multiple UAVs. Vehicles have to follow definite paths while in motion, the energy consumption of which can be calculated through integration of the product of thrust and vehicle velocity over the path. First of all, total region is divided into m number of regions with two different region division methods; URD (Uniform Region Division) and KVRD (K-means Clustering with Voronoi Region Division). Then, independent TSPs are solved for each of the m regions, and all TSPs solved in parallel using Genetic Algorithm (GA)(step 1-6) [62]. In step 7, I find the optimum number of UAVs, m, to have both the minimum fuel consumption and mission complete time. In step 8, I apply the Node Exchange Algorithm (NEA) to minimize the difference of the trajectory distances among m number of UAVs. In step 9, I finally apply UAV dynamic model to simulate real-time trajectories.

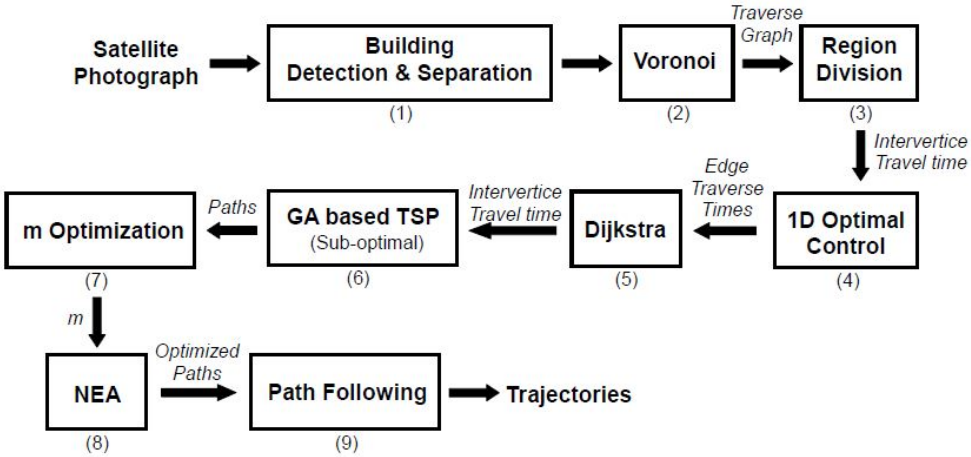
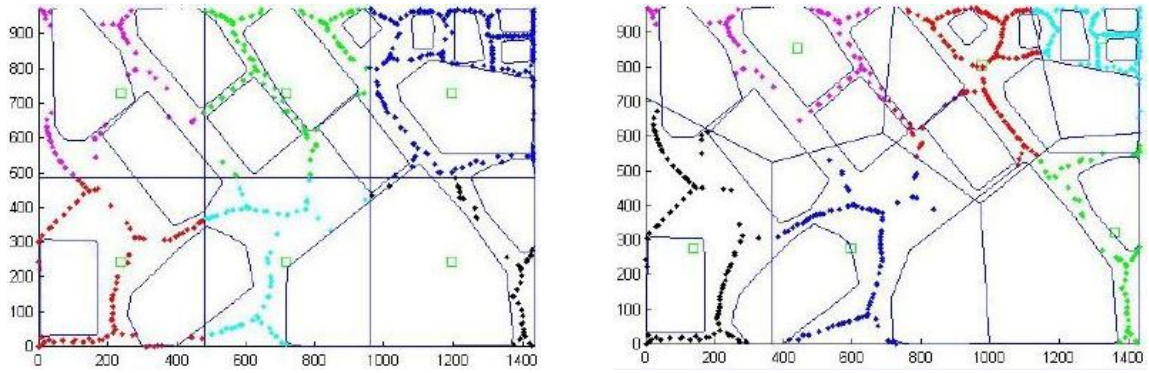


Figure 7.1.Hierarchy of mission planning.

In this chapter, I give two types of region division algorithms in Section 7.2, formulate the optimization problem in Section 7.3, descriptions on the NEA in Section 7.4, computational results in Section 7.5, and concluding remarks in Section 7.6.

7.2 Region Division Algorithm

To utilize multiple UAVs to do surveillance mission, I need to efficiently deploy UAVs over the interested area. In here, I assume that UAVs have homogeneous system properties. To maximize the fuel efficiency, each UAV should fly over the shortest distance without any overlapped searching area and it can be solved using mTSP. To achieve the goal, I apply region division methods to efficiently divide a given region into m number of regions using; Uniform Region Division method (URD) (Figure 7.2(a)); K-means Clustering algorithm [133] and Voronoi Region Division method (KVRD) (Figure 7.2(b)) [142]. Then, I send each UAV to each region to avoid possible collisions.



(a) URD method. (b) KVRD method.

Figure 7.2.Region division with URD and KVRD methods using six UAVs.

7.3 Optimization Problem Formulation

To equalize trajectories on the m number of regions, optimization cost function is introduced with some constraints. Using an assignment-based double-index integer programming formulation of mTSP can be given;

$$
\begin{aligned}
\min \quad & E\left[t_{mission} = \sum_{k=1}^{m}\left(\left(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ijk}x_{ijk}\right) + w_k + o_k\right)\right] && (7.1)\\
\text{s.t.} \quad & \sum_{p=1}^{m}\sum_{q=1,p\neq q}^{m} \left|t_p - t_q\right| < T_t,\\
& \sum_{j=2}^{n} x_{ijk} = 1 \quad \text{for any } k,\\
& \sum_{j=2}^{n} x_{jik} = 1 \quad \text{for any } k,\\
& \sum_{i\neq j} x_{ijk} = 1 \quad \text{for any } k,\\
& \sum_{i=j} x_{ijk} = 0 \quad \text{for any } k,\\
& c_{ijk} = G_t(i,j) \text{ for } i \neq j,\\
& w_k, o_k \geq 0,
\end{aligned}
$$

where, c_{ijk} is the time taken by kth UAV between ith and jth nodes, $x_{ijk} = \{0, 1\}$, n is the number of vertices, m is the number of UAVs, G_t is the time array among nodes $(n \times n)$, w_k is the time increase due to wind, o_k is the time increase due to unexpected obstacles, t_p is the time taken by pth UAV which is $(\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + w + o)_p$, and T_t is the threshold. The first constraint equation is used to result similar total distance of trajectories among m regions. Here, an appropriate threshold, T_t, is desired since the simulation time is getting longer as T_t is getting smaller.

Since the mission requires high fuel efficiency, I need to consider how to choose appropriate number of UAVs. To achieve this, I need to acknowledge that the fixed wing vehicle consumes more fuels when it takes off, climbs to certain altitude, and lands on the ground than when it flies at certain altitude. So, to save more fuels, less takeoff and landing is required. The total fuel consumption of a fixed wing vehicle can be expressed as,

$$n_f = F_t + F_c + F_n + F_l, \tag{7.2}$$

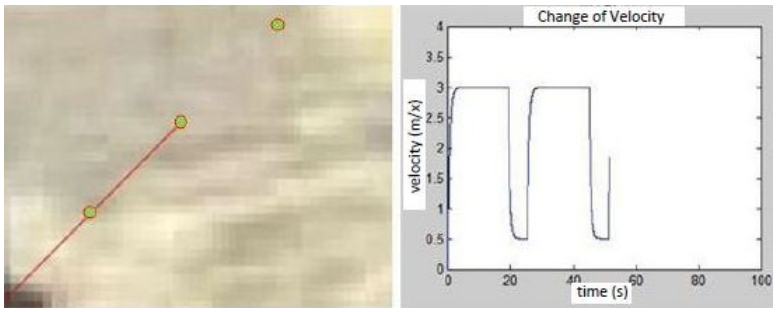
where n_f is the total number of fuel units burned, F_t is the fuel consumption during take off, F_c is the fuel consumption during increasing altitude, F_n is the fuel consumption when the fixed wing vehicle keeps at constant altitude, F_l is the fuel consumption during landing. With an assumption that the F_t, F_c, F_n, and F_l are constants, then Eq 7.2 can be rewritten as,

$$n_f = \dot{F}_b \left(c_t n_{t,t} + c_c n_{t,c} + c_n n_{t,n} + c_l n_{t,l} \right), \tag{7.3}$$

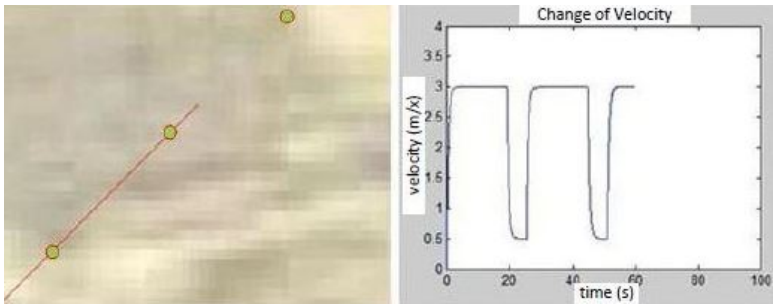
where c_t, c_c, c_n, and c_l are constants governing the fuel consumption rate which are dependent on the type of vehicle, $\dot{F}_b$ is the rate of fuel burn, and $n_{t,t}$, $n_{t,c}$, $n_{t,n}$, and $n_{t,l}$ are time taken for each function. If total m number of homogeneous UAVs are used, only c is changing with different total distance of given m trajectories. Then, Eq 7.3 becomes

$$\begin{aligned} N_f &= \dot{F}_b \left(j \left(c_t n_{t,t} + c_c n_{t,c} + c_l n_{t,l} \right) + c_n \sum_{i=1}^{j} n_{t_i,n} \right) < T_f, \quad \text{where, } j = \{1, ..., m\}, \\ N_t &= n_{t,t} + n_{t,c} + n_{t,l} + \max(n_{t_i,n}) < T_t, \quad \text{where, } i = \{1, ..., m\}, \end{aligned} \tag{7.4}$$

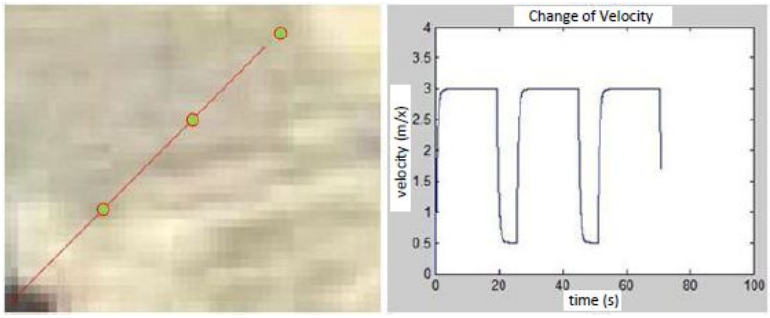
where N_f is the total amount of fuel consumption of m UAVs and N_t is the total flight time of m UAVs, that is, $N_f = [n_{f_1}, n_{f_2}, ..., n_{f_m}]$ and $N_t = [n_{t_1}, n_{t_2}, ..., n_{t_m}]$. Here, I introduce constraint equations, $N_f < T_f$ and $N_t < T_t$, since the amount of fuel which the UAV can carry and the time which the UAV should complete the mission are limited. The $n_{t_i,n}$ can be calculated using the UAV velocity profile (Fig 7.3) since the distance between two nodes are arranged with equalized distance. Due to the equalized distance between two nodes, UAVs have same flight time as $n_{t,t} + n_{t,c} + n_{t,l}$.



(a) Acceleration stage.



(b) Constant velocity stage.



(c) Deceleration stage.

Figure 7.3.Velocity profile of the UAV at acceleration, constant velocity, and deceleration stages (1D Optimal Control (Step4 in Fig 7.1)).

Since all homogeneous UAVs have same cycles, that is, UAVs have equal amount of time per each cycle as of $n_{t,t} + n_{t,c} + n_{t,l}$ and the traveling time of a trajectory can be expressed as,

$$n_{t,n} = (n_{edge} - 2)(n_{t,t} + n_{t,c} + n_{t,l}), \tag{7.5}$$

where n_{edge} is the total number of edges of the given trajectory and $n_{edge} - 2$ is used since it takes different amount of time at takeoff and landing stages. Then, Eq 7.4 becomes,

$$\begin{aligned} N_f &= \dot{F}_b \left(j \left(c_t n_{t,t} + c_c n_{t,c} + c_l n_{t,l} \right) + c_n \sum_{i=1}^{j} (n_{edge,i} - 2)(n_{t,t} + n_{t,c} + n_{t,l}) \right) \\ &< T_f, \end{aligned} \tag{7.6}$$

where $j = \{1, ..., m\}$. Also, to compensate the big difference in value between N_f and N_t due to the unit difference, I normalize those as,

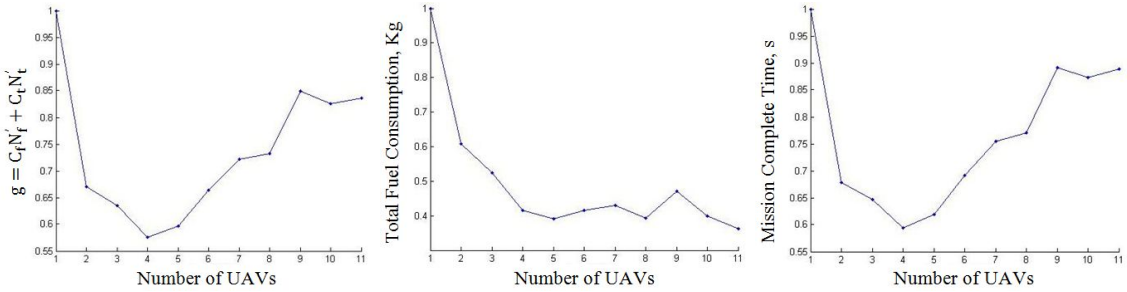
$$\begin{aligned} N'_f &= \frac{N_f}{\max(N_f)}, \\ N'_t &= \frac{N_t}{\max(N_t)}. \end{aligned} \tag{7.7}$$

Also, due to the fuel carriage limit, I flatten the total fuel consumption of each UAV by evenly dividing the surveillance area with the help of Node Exchange Algorithm (NEA) introduced in the following section. To achieve the best surveillance, I need to consider both fuel consumption and mission complete time. So, the cost function $g(m)$ can be constructed and I can get the desired number of UAVs, m, by minimizing $g(m)$ as

$$m = \arg\min \left[g(m) = C_f N'_f(m) + C_t N'_t(m) \right], \tag{7.8}$$

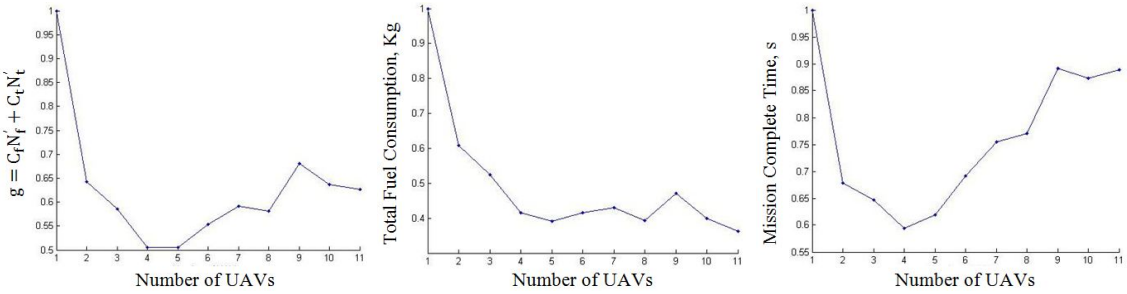
where C_f and C_t are relative costs of fuel consumption and mission complete time which are constrained by $C_f + C_t = 1$. Values of the C_f and C_t can be decided depends on the user preference on either fuel consumption or mission complete time. That is, if I put more emphasis on the less fuel consumption, I choose higher value for the C_f and vice versa. To choose the total number of UAVs, m, Eq 7.4 and Eq

7.8 are used to plot Fig 7.4 to Fig 7.6 by changing the coefficients, C_f and C_t by using $\dot{F}_b = 0.0067kg/s$, $c_t = 1.5$, $c_c = 1.3$, $c_n = 1$, $c_l = 1.2$, $n_{t,t} = 5s$, $n_{t,c} = 15s$, and $n_{t,l} = 10s$. When the coefficients are set as $C_f = 0.1$ and $C_t = 0.9$, I have the minimum cost, g, at four UAVs. When the coefficients are set as $C_f = 0.5$ and $C_t = 0.5$, I have the minimum cost, g, at four and five UAVs. When the coefficients are set as $C_f = 0.9$ and $C_t = 0.1$, I have the minimum cost, g, with five UAVs. The choice of the coefficient values depends upon the mission purposes and I choose the usage of four UAVs through the rest of paper.



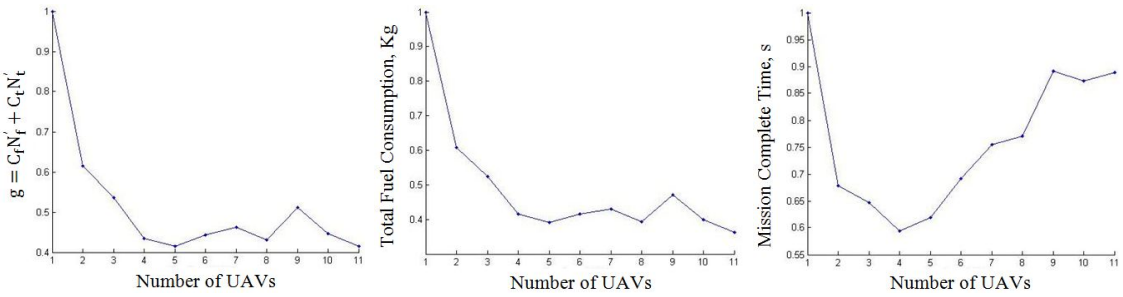
(a) Number of UAVs Vs. $g(m)$ in Eq 7.8. (b) Number of UAVs Vs. total fuel consumption. (c) Number of UAVs Vs. mission complete time.

Figure 7.4.Application of Eq 7.4 and Eq 7.8 to choose the total number of UAVs (when $C_f = 0.1$ and $C_t = 0.9$).



(a) Number of UAVs Vs. $g(m)$ in Eq 7.8. (b) Number of UAVs Vs. total fuel consumption. (c) Number of UAVs Vs. mission complete time.

Figure 7.5.Application of Eq 7.4 and Eq 7.8 to choose the total number of UAVs (when $C_f = 0.5$ and $C_t = 0.5$).

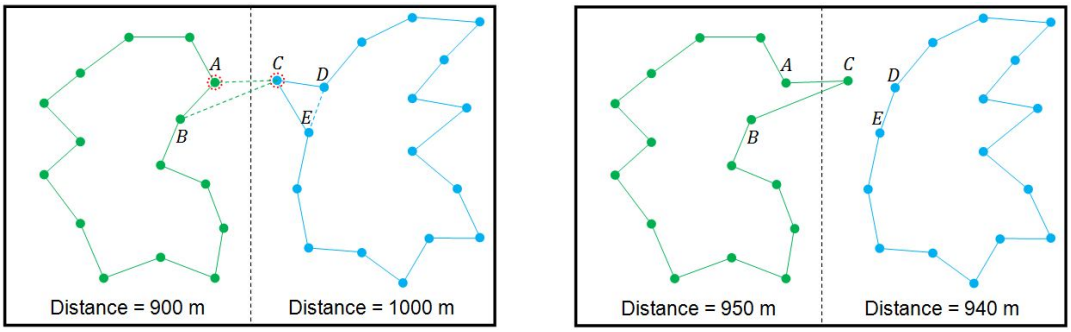


(a) Number of UAVs Vs. $g(m)$ in Eq 7.8. (b) Number of UAVs Vs. total fuel consumption. (c) Number of UAVs Vs. mission complete time.

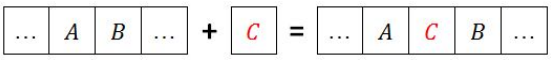
Figure 7.6.Application of Eq 7.4 and Eq 7.8 to choose the total number of UAVs (when $C_f = 0.9$ and $C_t = 0.1$).

7.4 Node Exchange Algorithm (NEA)

Even if UAVs carry equal amount of fuels, fuel efficiency might degrade if total distance of each trajectory is not equalized among UAVs since some UAVs fly longer and some UAVs fly less. Trajectory equalization using NEA can solve this problem and this process can be applied by giving and taking the adjacent nodes among m trajectories (Fig 7.7). The processes of NEA is explained in the below.



(a) Before NEA is applied. (b) After NEA is applied.



(c) NEA operation.

Figure 7.7.Procedure of NEA.

1. Pick up the closest two nodes among nodes in the longest trajectory and nodes in its adjacent trajectories. I assume those two nodes are A and C.

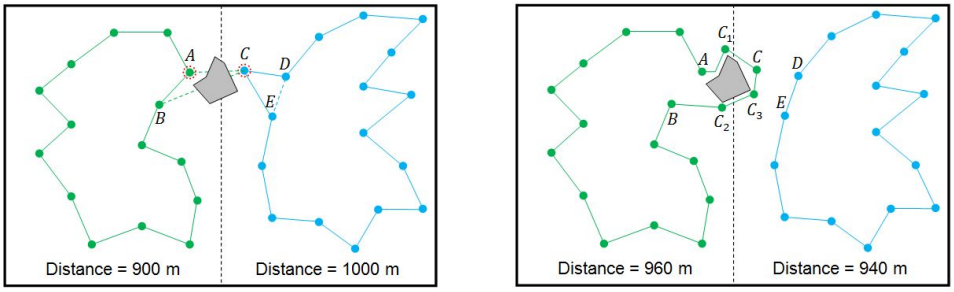
2. Move node C in the longest trajectory to the adjacent trajectory which contains the closest node from the longest trajectory. Then, re-run TSP.

3. If step 1 and step 2 fails to find two nodes due to local minimum, pick up the closest two nodes among nodes in the shortest trajectory and nodes in its adjacent trajectories.

4. Continue step 1 to step 3 until the difference between the maximum and minimum trajectory distances becomes less than some threshold T_d.

That is, every adjacent two trajectories keeps comparing and exchanging nodes until the difference between two total distances become less than T_d. Often times, new trajectories after the nodes exchange is applied cross neighbor blocks and so I need to apply the Path Sliding Algorithm (PSA) to avoid collisions by letting trajectories to go along the block with an assumption that UAV can only detour blocks because of the height of the buildings (Fig 7.8). Due to the unexpected shape of the encountered block, there are additional distances in the amount of $\overline{A, C, B} - \overline{A, C_1, C, C_3, C_2, B}$ (Fig 7.8(b)). The size of the buffer zone is discussed in paper [62] so I skip the description. When PSA determined a trajectory along the building, there appears two candidates trajectories and I choose the shorter trajectory. Oftentimes, I encounters a shorter trajectory which UAV already flied before. In that case, I choose the longer trajectory to avoid overlaps of the trajectories.

This method can be applied to the Fig 7.2(a). To search the given region, it is not necessary to visit all nodes as shown in Fig 7.2(a), so I minimize the number of nodes by selecting the coincident points of three vectors (Fig 7.9 and Fig 7.10). In Fig 7.10(a), red and green circles represent the nodes which are exchanged between two UAVs to equalize the total trajectory distances. Here, the number at each region represents the order of UAVs.

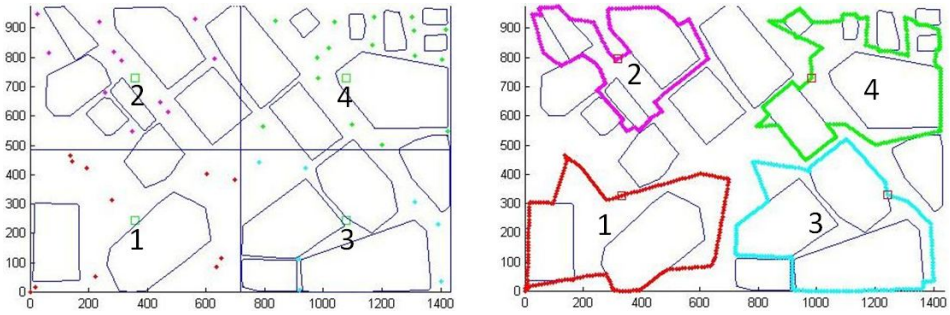


(a) Before PSA is applied.

(b) After PSA is applied.

(c) PSA operation.

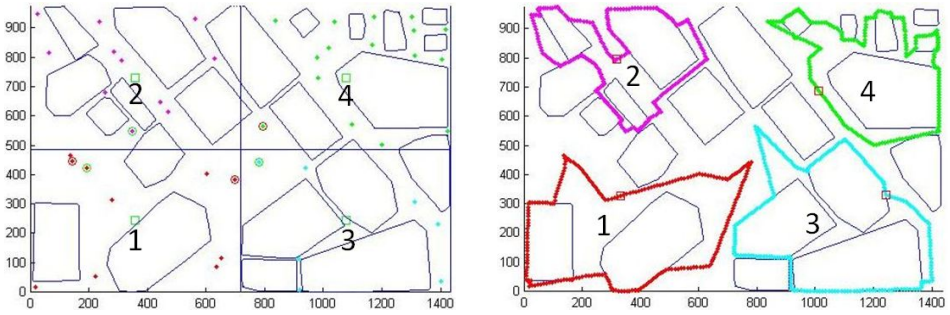
Figure 7.8.Procedure of PSA.



(a) Total node before NEA is applied.

(b) Final trajectories before NEA is applied.

Figure 7.9.Before NEA is applied using four UAVs with regions divided with URD (unit: m).



(a) Total node after NEA is applied.

(b) Final trajectories after NEA is applied.

Figure 7.10.After NEA is applied using four UAVs with regions divided with URD (unit: m).

7.5 Computational Results

I use tracking models of UAVs from prior work [117]. These models are accurate approximations to the dynamics of attitude stabilized vehicles. This is because global or semi-global closed loop stabilization transforms UAV dynamics into target dynamics of point mass models with definite tracking time constants for position, τ_x and velocity, τ_v. Path following aircraft dynamics which is dominated by the slowest or dominant poles is far more accurate than simplistic point mass or constant velocity models. Second, these models also take into account the time delays of communication and computation involved in tracking targets or prey vehicles. The UAV tracking model is defined as,

$$\begin{aligned} \mathbf{x}_c(k+1) &= \mathbf{x}_c(k) + T\mathbf{v}_c(k), \\ \mathbf{v}_c(k+1) &= -\frac{T}{\tau_x\tau_v}\mathbf{x}_c(k) + \left(1 - \frac{T}{\tau_v}\right)\mathbf{v}_c(k) + \frac{T}{\tau_x\tau_v}\mathbf{x}_c^{ref}(k), \end{aligned} \tag{7.9}$$

where T is the sampling time, x_p is the planar position of prey $[x_{px}, x_{py}]$, v_p is the velocity vectors of prey, x_c is the tree dimensional position of chaser $[x_{cx}, x_{cy}, x_{cz}]$, v_c the velocity vectors of chaser, and x_c^{ref} is the tracking set point. All the simulations are done with a fixed wing vehicle which has following system properties; $Velocity_{min} = 5m/s$, $Velocity_{max} = 10m/s$, initial velocity as $[0, 0, 0]m/s$, $Acceleration_{max} = 5m/s$, $Altitude_{min} = 25m$, $Altitude_{max} = 50m$, $Altitude_{normal} = 30m$, time step of Simulink as $0.01s$, $\tau_x = 0.25s$, and $\tau_v = 0.5s$. All simulations are done using a desktop with Intel(R) Core(TM) 2 Duo CPU 3.00 GHz Processor, 64-bit Operating System, and 4.00 GB RAM. Flight trace of Fig 7.10(b) is simulated as in Fig 7.11. Red rectangle represents the starting location of UAVs and Red circle represents the ending locations of UAVs. There are little bit of ripples along the trajectories largely due to the time constants, τ_x and τ_v, but UAVs generally fly well along the predetermined trajectories. Also, those flight ripples does not cause any collision on UAVs since I included buffer zones around the blocks large enough by concerning the maximum

velocity and turning angle of UAVs [62]. Those ripples will be decreased if I use hover capable UAVs instead which have zero minimum velocity. From the simulated trajectories in Fig 7.11, I can predict the approximate traveling time and the amount of fuel consumption of each UAV.

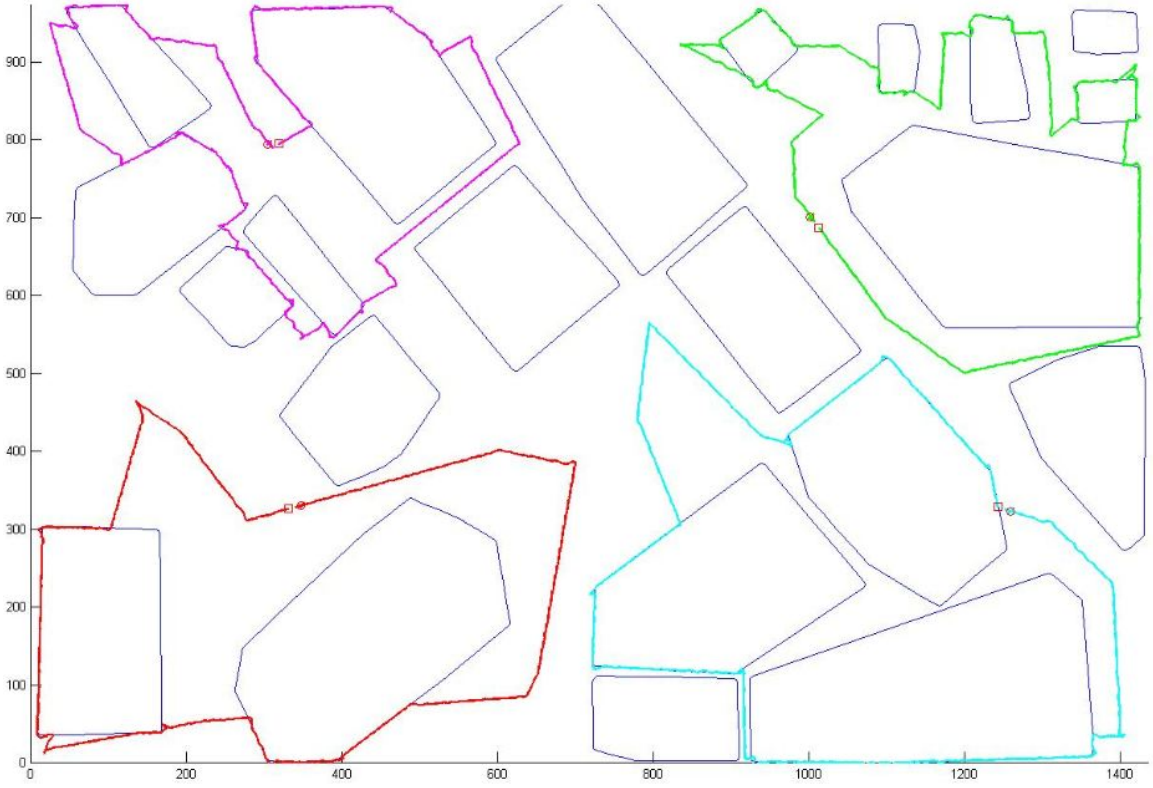
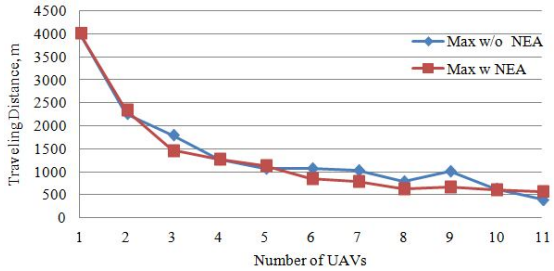


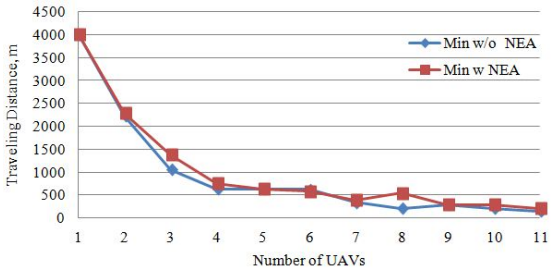
Figure 7.11.Flight traces of four UAVs (unit: m).

Now, NEA is applied to different number of UAVs to see the effects on the max, min, max-min, and total distances. In the aspect of the maximum and the minimum traveling distances, the case without NEA has larger maximum distance than the one with NEA (Fig 7.12(a)) and the case without NEA has lower minimum distance than the one with NEA (Fig 7.12(b)). Also, the case with NEA has much less $Max - Min$ value compare to the case when NEA is not applied (Fig 7.12(c)). This result shows that the NEA processes indeed flatten the differences among m trajectories. In addition, I should acknowledge the fact that the total traveling distances without

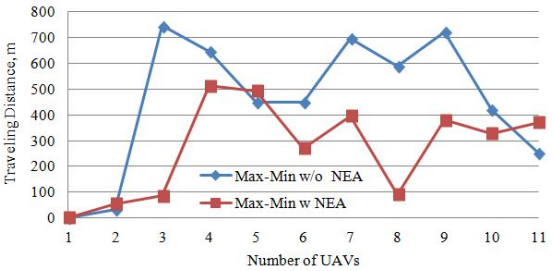
NEA and with NEA do not have major difference which indicates that NEA does not always bring increased trajectories (Fig 7.12(d)).



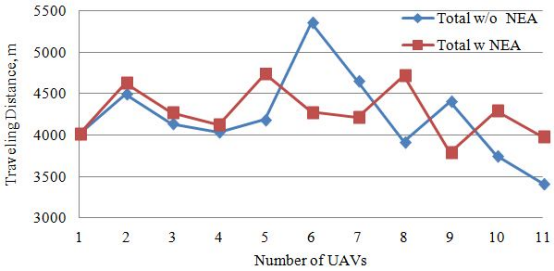
(a) The *max* traveling distance changes without and with NEA.



(b) The *min* traveling distance changes without and with NEA.



(c) The $max - min$ traveling distance without and with NEA.



(d) The *total* traveling distance without and with NEA.

Figure 7.12.NEA application on different number of UAVs.

7.6 Concluding Remarks

I have shown the methods of achieving the maximum fuel efficiency when I perform surveillance missions by converting mTSP to m TSPs based on GA with the optimized number of UAVs, m. The optimized value m will be different depends on the missions (minimum traveling time, minimum fuel consumption, etc) and it can be controlled by choosing proper weights, C_f and C_t. To compensate the achievement of the minimum fuel consumption, NEA is developed to equalize the total traveling distances among UAVs and PSA is developed to prevent any possible collision of UAVs. Once I have m number of equalized trajectories, I can send m homogeneous UAVs to each region. This method is generally much safer surveillance mission planning than the swarming

surveillance mission planning in the sense that each UAVs have the least number of coming across. In addition, my work opens up several possibilities for both system and algorithm development:

1. Come up with more realistic relationships between fuel consumption and UAV vehicle dynamics,

2. Present a comparison table about fuel consumption rate between normal mTSP trajectories and mTSP trajectories with increased fuel efficiency.

CHAPTER 8: APPLICATION–CATTLE ROUNDUP

Two quadrotor UAVs are maneuvered to guide two animals into their pen within the minimum time by creating noise modeled with an exponential function. The quadrotor UAVs are stabilized and controlled via feedback linearization and follow optimal trajectories by avoiding collisions based on the dynamic programming.

8.1 Background and Motivation

Beyond a simple mission of target detection [142], some missions require to track multiple targets by a fleet of UAVs and UGVs. Tracking targets using the cooperation of UAVs and UGVs were solved in other works using a probabilistic game theoretical framework [143], a gradient search approach and the probabilistic threat exposure map (PTEM) [144], the classic Homicidal Chauffeur problem [145], and a framework combining game theory and geometry [146].

Quadrotors have become popular due to the feasibility of their use in various applications. With this trend, control engineers designed many kinds of controllers (backstepping [147–151], Feedback Linearization [152–155], PD [156, 157]) for the quadrotor.

Increased autonomy of quadrotors can automate the grazing cattle by controlling the movement of the heard of cattle and optimizing the pasture growth at the same time. Quadrotors can carry a small size speaker which makes some noises of predators to repel animals to the certain direction.

I formulate the dynamic models of the UAV and prey in Section 8.2 and the attitude and translation controllers of UAV using the feedback linearization controller in Section 8.3. Mission scenario optimization using two UAVs and preys is discussed in Section 8.4 and the collision avoidance algorithms of UAVs and preys are described

in the optimization sense in Section 8.5. I show simulation results in Section 8.6 and conclude in Section 8.7 with possible future works.

8.2 UAV and Prey Dynamic Model

8.2.1 UAV Dynamic Model

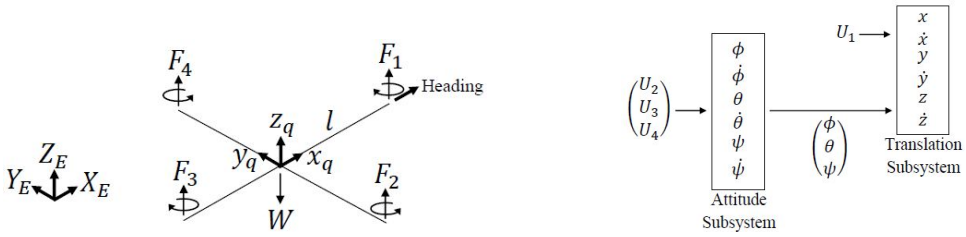
Quadrotor UAV dynamics are derived in [64] as

$$\begin{aligned}
\dot{V} &= \begin{bmatrix} \dot{V}_1 \\ \dot{V}_2 \\ \dot{V}_3 \end{bmatrix} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} (c\phi s\theta c\psi + s\phi s\psi)\frac{1}{m} \\ (c\phi s\theta s\psi - s\phi c\psi)\frac{1}{m} \\ (c\phi c\theta)\frac{1}{m} \end{bmatrix} U_1 + \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix}, \qquad (8.1) \\
\dot{w} &= \begin{bmatrix} \dot{w}_1 \\ \dot{w}_2 \\ \dot{w}_3 \end{bmatrix} = \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \begin{bmatrix} \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J}{I_x}\dot{\theta}\Omega + \frac{l}{I_x}U_2 \\ \dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) - \frac{J}{I_y}\dot{\phi}\Omega + \frac{l}{I_y}U_3 \\ \dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \frac{l}{I_z}U_4 \end{bmatrix},
\end{aligned}$$

where x, y, and z are the UAV position, ϕ, θ, and ψ are the roll, pitch, and yaw, $c\theta$ and $s\theta$ represent $\cos\theta$ and $\sin\theta$, $I_{x,y,z}$ is body inertias, J is a propeller inertia, and l is a lever (i.e., propeller length). The system's inputs (U_1, U_2, U_3, U_4 and Ω) can be rewritten as

$$\begin{aligned}
U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2), \qquad (8.2) \\
U_2 &= b(\Omega_4^2 - \Omega_2^2), \\
U_3 &= b(\Omega_3^2 - \Omega_1^2), \\
U_4 &= d(\Omega_2^2 + \Omega_4^2 - \Omega_1^2 - \Omega_3^2), \\
\Omega &= \Omega_2 + \Omega_4 - \Omega_1 - \Omega_3,
\end{aligned}$$

where Ω_i is a rotor speed, b is a thrust factor, and d is a drag factor. The configuration and governing control inputs of the quadrotor are described in Figure 8.1.



(a) Quadrotor configuration.

(b) Governing controller structure of the quadrotor.

Figure 8.1.Quadrotor configuration and governing control inputs.

8.2.2 Prey Dynamic Model

Prey dynamics during the normal period (when UAVs are away from the preys) are assumed as

$$\begin{bmatrix} x \\ y \end{bmatrix}_{i+1} = \begin{bmatrix} x + cvt\cos(\theta_n) \\ y + cvt\sin(\theta_n) \end{bmatrix}_i, \tag{8.3}$$

and during the emergency period (when UAVs are near the preys) as

$$\begin{bmatrix} x \\ y \end{bmatrix}_{i+1} = \begin{bmatrix} x + cvt\cos(\theta_e) \\ y + cvt\sin(\theta_e) \end{bmatrix}_i, \tag{8.4}$$

where x_i and y_i are the position of the prey at the ith step, c is a constant between 0 and 1 which is dependent on the distance between the UAV and prey, v is the speed of the prey, t is the step time, θ_n is the heading direction of the prey during the normal period, and θ_e is the heading direction of the prey during the emergency period. The

θ_n is randomly chosen at every step, but the θ_e is decided to let the prey move in the opposite direction from the UAV as $\theta_e = \theta_{UAV} + \pi + f$ where $-\frac{\alpha}{2} \leq f \leq \frac{\alpha}{2}$ and α is the maximum tilt angle of the prey.

The UAV flies to the point, P_e, which is located on the line along the center of the cage and the prey to repel the prey to the cage (Figure 8.2) by creating noises. Due to the randomized loudness of sounds and variation of speeds approaching the stocks of UAVs (i.e., behaviour like a sheepdog), the animals they will not be able to adapt to the motion of the UAVs around them and herding should work. The strength of the noises can be modeled with an exponential function as

$$s(L) = h\left(\frac{e^{-L} - e^{-b}}{1 - e^{-b}}\right) + s_r, \tag{8.5}$$

where L is the distance between the UAV and prey, b is the radius of the noise effective area, h is the magnitude of the maximum noise, and s_r is the random noise sound based on the normal distribution $N(0, 0.01h^2)$. Equation 8.5 guarantees that I have the maximum noise, h, at the center of the UAV and zero noise at the boundary of the effective area. Then, the c in Equation 8.3 will be

$$\begin{aligned} c(L) &= a, && \text{if} \quad L > b, \\ &= a + (1-a)\left(\frac{e^{-L} - e^{-b}}{1 - e^{-b}}\right), && \text{if} \quad 0 \leq L \leq b, \end{aligned} \tag{8.6}$$

where a is the constant governing the speed of the prey during the normal period (< 0.5). The c is low during the normal period and getting closer to 1 as the prey is getting closer to the UAV.

Then, Equation 8.3 and Equation 8.4 become

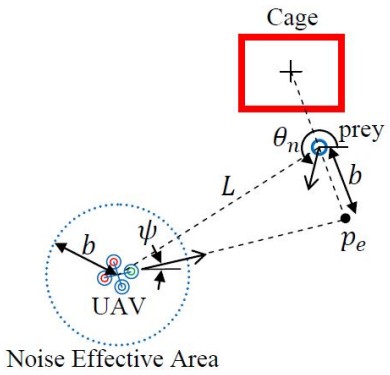


Figure 8.2.Noise effective area of the UAV.

$$\begin{bmatrix} x \\ y \end{bmatrix}_{i+1} = \begin{bmatrix} x + avt\cos(\theta_n) \\ y + avt\sin(\theta_n) \end{bmatrix}_i, \quad \text{if} \quad L > b, \tag{8.7}$$

$$\begin{bmatrix} x \\ y \end{bmatrix}_{i+1} = \begin{bmatrix} x + \left(a + (1-a)\left(\frac{e^{-L}-e^{-b}}{1-e^{-b}}\right)\right) vt\cos(\theta_e) \\ y + \left(a + (1-a)\left(\frac{e^{-L}-e^{-b}}{1-e^{-b}}\right)\right) vt\sin(\theta_e) \end{bmatrix}_i, \quad \text{if} \quad 0 \le L \le b.$$

8.3 UAV Controller Design

8.3.1 Attitude Controller

I can control the roll (ϕ) using the Feedback Linearization controller. From Equation 8.1,

$$\ddot{\phi} = \dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) - \frac{J}{I_x}\dot{\theta}\Omega + \frac{l}{I_x}U_2, \tag{8.8}$$

and rearrange it as,

$$U_2 = \frac{I_x}{l}\left[-\dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) + \frac{J}{I_x}\dot{\theta}\Omega + \nu_2\right], \tag{8.9}$$

where ν_2 is a tuning function. Then, $\ddot{\phi}$ becomes,

$$\ddot{\phi} = \nu_2 \triangleq \ddot{\phi}_d - 2\lambda_2\dot{e} - \lambda_2^2 e, \tag{8.10}$$

where $\ddot{\phi}_d$ is the desired state of $\ddot{\phi}$ and λ_2 is a positive tuning constant. By rearranging, Equation 8.10 becomes

$$\begin{aligned} \left(\ddot{\phi} - \ddot{\phi}_d\right) + 2\lambda_2\dot{e} + \lambda_2^2 e &= 0, \\ \ddot{e} + 2\lambda_2\dot{e} + \lambda_2^2 e &= 0. \end{aligned} \tag{8.11}$$

Now, let the Lyapunov function V as $V = \frac{p_2}{2}e^2 + \frac{q_2}{2}\dot{e}^2$ where p_2 and q_2 are positive tuning constants which result V to be locally positive definite about $\ddot{\phi}_d$. Then $\dot{V}$ becomes

$$\begin{aligned} \dot{V} &= p_2 e\dot{e} + q_2\dot{e}\ddot{e}, \\ &= p_2 e\dot{e} + q_2\dot{e}\left(-2\lambda_2\dot{e} - \lambda_2^2 e\right), \\ &= p_2 e\dot{e} - 2\lambda_2 q_2\dot{e} - q_2\lambda_2^2 e\dot{e}, \\ &= e\dot{e}\left(p_2 - q_2\lambda_2^2\right) - 2\lambda_2 q_2\dot{e}^2. \end{aligned} \tag{8.12}$$

If I choose $\lambda_2 = \sqrt{\frac{p_2}{q_2}} > 0$, Equation 8.12 becomes

$$\dot{V} = -2\lambda_2 q_2\dot{e}^2 < 0, \tag{8.13}$$

and it proves that the state $\ddot{\phi}$ converges asymptotically to the state $\ddot{\phi}_d$ according to the Lyapunov stability theorem [158]. Also, from Equation 8.9 and Equation 8.10, I can achieve the control input U_2 as

$$
\begin{aligned}
U_2 &= \frac{I_x}{l}\left[-\dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) + \frac{J}{I_x}\dot{\theta}\Omega + \ddot{\phi}\right], \\
&= \frac{I_x}{l}\left[-\dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) + \frac{J}{I_x}\dot{\theta}\Omega + \ddot{\phi}_d - 2\lambda_2\dot{e} - \lambda_2^2 e\right], \\
&= \frac{I_x}{l}\left[-\dot{\theta}\dot{\psi}\left(\frac{I_y - I_z}{I_x}\right) + \frac{J}{I_x}\dot{\theta}\Omega + \ddot{\phi}_d - 2\lambda_2\left(\dot{\phi} - \dot{\phi}_d\right) - \lambda_2^2(\phi - \phi_d)\right].
\end{aligned} \tag{8.14}
$$

In the same way, U_3 and U_4 can be calculated as

$$
\begin{aligned}
U_3 &= \frac{I_y}{l}\left[-\dot{\phi}\dot{\psi}\left(\frac{I_z - I_x}{I_y}\right) + \frac{J}{I_y}\dot{\phi}\Omega + \ddot{\theta}_d - 2\lambda_3\left(\dot{\theta} - \dot{\theta}_d\right) - \lambda_3^2(\theta - \theta_d)\right], \\
U_4 &= \frac{I_z}{l}\left[-\dot{\phi}\dot{\theta}\left(\frac{I_x - I_y}{I_z}\right) + \ddot{\psi}_d - 2\lambda_4\left(\dot{\psi} - \dot{\psi}_d\right) - \lambda_4^2(\psi - \psi_d)\right],
\end{aligned} \tag{8.15}
$$

with additional positive tuning constants, λ_3 and λ_4.

8.3.2 Translation Controller

Since $\ddot{x}$, $\ddot{y}$, and $\ddot{z}$ are simultaneously controlled by the U_1, I introduce $r = \sqrt{x^2 + y^2 + z^2}$. By differentiating with respect to time, I can achieve

$$
\begin{aligned}
\dot{r} &= \frac{x\dot{x} + y\dot{y} + z\dot{z}}{\sqrt{x^2 + y^2 + z^2}}, \\
\ddot{r} &= -\frac{(x\dot{x} + y\dot{y} + z\dot{z})^2}{(x^2 + y^2 + z^2)^{\frac{3}{2}}} + \frac{\dot{x}^2 + \dot{y}^2 + \dot{z}^2 + x\ddot{x} + y\ddot{y} + z\ddot{z}}{\sqrt{x^2 + y^2 + z^2}}.
\end{aligned} \tag{8.16}
$$

By substituting $\ddot{x}$, $\ddot{y}$, and $\ddot{z}$ from Equation 8.1 to Equation 8.16, I can get

$$\begin{aligned}\ddot{r} &= -\frac{\left(x\dot{x}+y\dot{y}+z\dot{z}\right)^2}{\left(x^2+y^2+z^2\right)^{\frac{3}{2}}}+\frac{1}{\sqrt{x^2+y^2+z^2}}\Bigg(\frac{U_1(c\phi c\psi s\theta+s\phi s\psi)x}{m} \\ &\quad +\frac{U_1(-c\psi s\phi+c\phi s\theta s\psi)y}{m}+\left(-g+\frac{U_1 c\theta c\phi}{m}\right)z+\dot{x}^2+\dot{y}^2+\dot{z}^2\Bigg).\end{aligned} \tag{8.17}$$

By rearranging, I get

$$\ddot{r} = AU_1 + B, \tag{8.18}$$

where,

$$\begin{aligned}A &= -\frac{1}{m\sqrt{x^2+y^2+z^2}}\left((c\phi c\psi s\theta+s\phi s\psi)x+(-c\psi s\phi+c\phi s\theta s\psi)y+(c\theta c\phi)z\right), \\ B &= \frac{1}{\sqrt{x^2+y^2+z^2}}\left(-gz+\dot{x}^2+\dot{y}^2+\dot{z}^2\right)+\frac{\left(x\dot{x}^2+y\dot{y}^2+z\dot{z}^2\right)^2}{\left(x^2+y^2+z^2\right)^{\frac{3}{2}}}.\end{aligned}$$

Then, I apply the same method of the Lyapunov stability proof as before and it introduces additional positive tuning constants, λ_1. In addition, from Equation 8.18, I can achieve the control input as

$$\begin{aligned}U_1 &= -\frac{B}{A}+\nu_1, \\ &= -\frac{B}{A}+\ddot{r}_d-2\lambda_1\dot{e}-\lambda_1^2 e, \\ &= -\frac{B}{A}+\ddot{r}_d-2\lambda_1\left(\dot{r}-\dot{r}_d\right)-\lambda_1^2\left(r-r_d\right), \\ &= -\frac{B}{A}+A_dU_1+B_d-2\lambda_1\left(\frac{x\dot{x}+y\dot{y}+z\dot{z}}{\sqrt{x^2+y^2+z^2}}-\frac{x_d\dot{x}_d+y_d\dot{y}_d+z_d\dot{z}_d}{\sqrt{x_d^2+y_d^2+z_d^2}}\right) \\ &\quad -\lambda_1^2\left(\sqrt{x^2+y^2+z^2}-\sqrt{x_d^2+y_d^2+z_d^2}\right).\end{aligned} \tag{8.19}$$

By rearranging, I can get

$$\begin{aligned} U_1 &= \frac{1}{1-A_d}\Bigg(-\frac{B}{A} + B_d - 2\lambda_1 \left(\frac{x\dot{x}+y\dot{y}+z\dot{z}}{\sqrt{x^2+y^2+z^2}} - \frac{x_d\dot{x}_d+y_d\dot{y}_d+z_d\dot{z}_d}{\sqrt{x_d^2+y_d^2+z_d^2}} \right) \\ &\quad -\lambda_1^2 \left(\sqrt{x^2+y^2+z^2} - \sqrt{x_d^2+y_d^2+z_d^2} \right) \Bigg). \end{aligned} \tag{8.20}$$

8.4 Mission Scenario

I consider two UAVs chasing two preys throughout the chapter. This can be expanded to a lager number of preys and UAVs. I assume here that the entire group of vehicles has only a single objective of minimizing the expected time to complete a mission which is guiding multiple preys to the cage using multiple UAVs. In mathematical programming language:

$$\begin{aligned} &\min \quad E\left[t_{mission}\right], \\ &\text{s.t.} \begin{pmatrix} \text{UAV dynamic constraints} \\ \text{UAV sensor and actuator constraints} \\ \text{UAV collision avoidance constraints} \\ \text{Prey dynamic constraints} \\ \text{Prey motion constraints} \\ \text{Prey collision avoidance constraints} \end{pmatrix}, \end{aligned} \tag{8.21}$$

where $E\left[t_{mission}\right]$ is the expected time required for the mission and all of the constraints are stochastic in nature. The mission scenario configuration is shown in Figure 8.3 where GS_1 and GS_2 are two ground stations where preys initiate, d is the distance from the ground stations to the UAVs, L is the distance from the UAVs to the preys, and l is the distance from the preys to the cage. The total distance of UAV movement $d_{UAV_i,total} = d_i + L_i + l_i$ and UAVs need to strategically decide which

combination to use between $L_{i,1}+l_1$ and $L_{i,2}+l_2$. The mission requirement is to guide two preys to the cage within the minimum time, so all four distance combination of both UAVs need to be simultaneously taken into account.

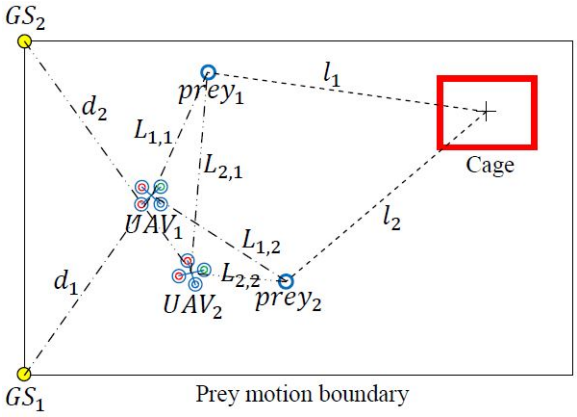
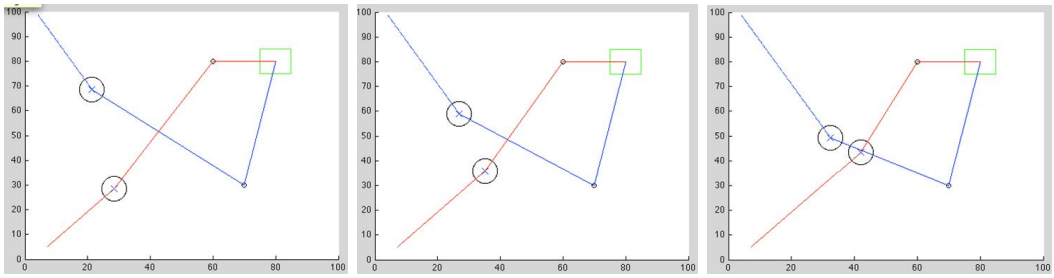


Figure 8.3.Mission scenario configuration.

The problem size, i.e. 2 UAVs handling two cattle, can be augmented to the increased number of UAVs for larger herds by introducing equations for the motion of a herd that generally follows leaders which is how a sheepdog or a cowboy handles a herd. The UAVs can round the herd and move them to the pen by chasing the leaders in the right direction, and then circling the herd.

8.5 Collision Avoidance

Since multiple UAVs and preys are involved in the mission, I need to design a collision avoidance algorithm. To avoid a collision between UAVs, new constraints are needed to be added into the algorithm, and cost function should be modified. In this chapter, dynamic programming optimization tool is used to minimize the total mission planning time because dynamic programming is a powerful technique for a sequence of decisions. Dynamic programming reduces the runtime of algorithm avoiding calculation of same subproblem and numerically finds a good feasible solution close to optimal. The numerical accuracy of the solution does not matter here beyond



(a) Step 1. (b) Step 2. (c) Step 3.

Figure 8.4.Collision avoidance simulation.

a point (collision avoidance) since I do not want the UAVs to get too close otherwise wind gusts could cause collisions.

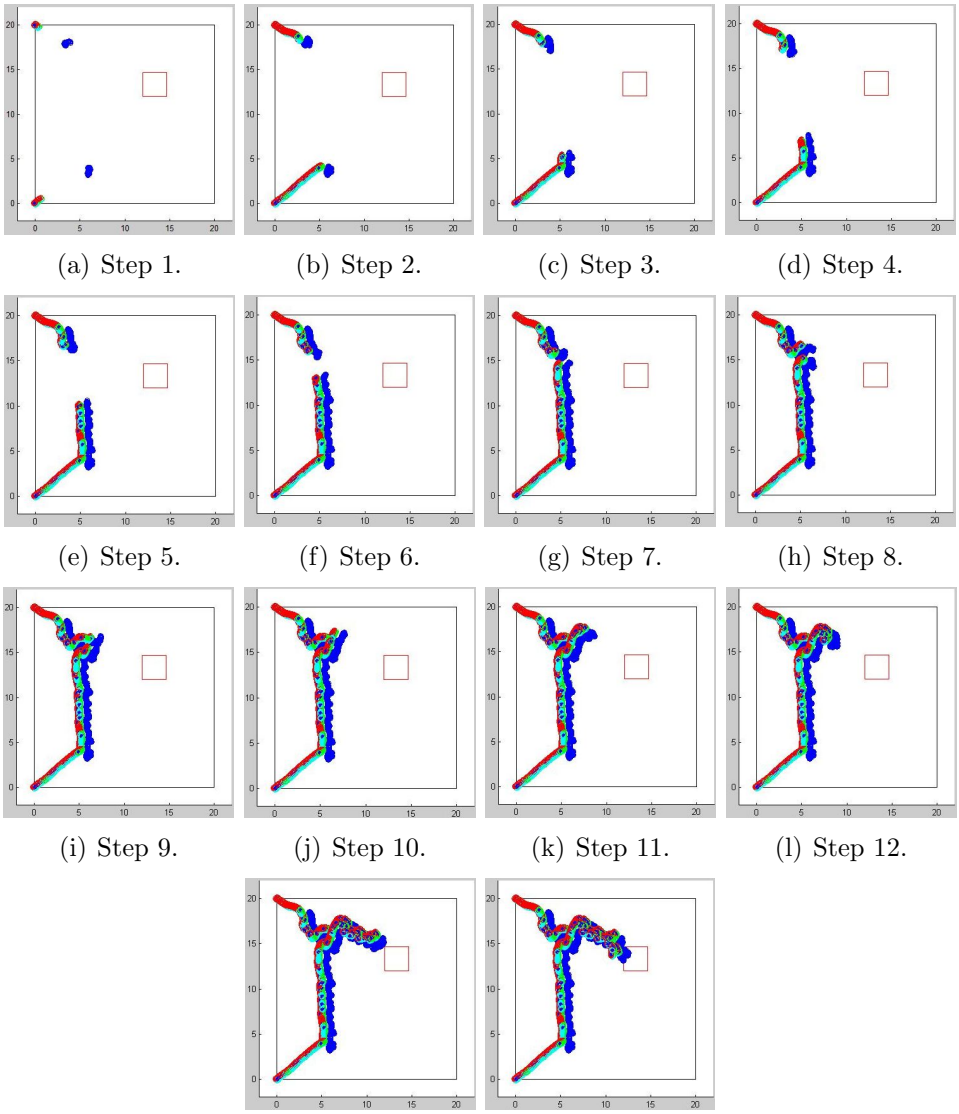
$$\min \quad E\left[J=\sum_{i=1}^{m}\sum_{j=1}^{n}(j+\parallel \theta_{uav}-\theta_{d,uav} \parallel)_i\right], \tag{8.22}$$

$$\text{s.t.} \begin{bmatrix} x_{uav} \\ y_{uav} \end{bmatrix}_{i+1} = \begin{bmatrix} x_{uav} \\ y_{uav} \end{bmatrix}_{i} + Tv_{uav} \begin{bmatrix} \cos(\theta_{uav}) \\ \sin(\theta_{uav}) \end{bmatrix}_{i},$$

where, m is the total number of UAVs, n is the total iterations, $\theta_{d,uav}=\frac{y_{prey}-y_{uav}}{x_{prey}-x_{uav}}$, T is the sampling time, $v_{min,uav} \leq v_{uav} \leq v_{max,uav}$, $\parallel p_{uav,1}-p_{uav,2} \parallel \geq R_{uav}^2$, p_{uav} is the position of UAV, and R_{UAV} is the minimum separation distance to avoid collisions between UAVs. To reduce algorithmic run-time, in real-time operation, I stop iterating once I reach a feasible or sub-optimal solution (Fig 8.4). The algorithmic complexity increases with the size of the herd and the number of UAVs.

8.6 Simulation

The size of the prey motion boundary is set to be 20 by 20 and the cage is set at $(13.33, 13.33)$ with the size 1.33. UAV_1 takes off at $(0, 0, 0)$, UAV_2 takes off at $(0, 20, 0)$, Prey_1 leaves at $(6, 4, 0)$, and Prey_2 leaves at $(4, 18, 0)$. Figure 8.5 shows the simulation at each time step using SIMULINK® (Appendix F.).



(a) Step 1. (b) Step 2. (c) Step 3. (d) Step 4.

(e) Step 5. (f) Step 6. (g) Step 7. (h) Step 8.

(i) Step 9. (j) Step 10. (k) Step 11. (l) Step 12.

(m) Step 13. (n) Step 14.

Figure 8.5.Simulation of UAVs chasing preys.

8.7 Concluding Remarks

I have shown that the strategic guidance of two preys using two UAVs with the nonlinear optimization tool and the feedback linearization controller for the UAV. My work opens up several possibilities for both system and algorithm developments:

1. Gathering cattle to multiple locations using multiple UAVs with the strategic optimization (i.e. I group cattle and one UAV is continuously circling them, they will stay together while the other UAV drives other individuals or groups to the main herd. For larger herds, I can have more than 1 UAV circling the herd so it stays together.),

2. Development of the HILS (Hardware-in-the-loop Simulation) to demonstrate the feedback linearization controller.

PART 4: CONCLUSION AND FUTURE WORK

CHAPTER 9: CONCLUSION AND FUTURE WORK

The thesis starts with the history of UAV ranges from the middle of nineteenth century to now. I also show six major functions of UAV listed as target & decoy, reconnaissance, combat, logistics, and civil & commercial UAVs. Then, I analyze uncertainties on position, orientation, battery consumption, and trajectory generation and study methods to mitigate each of the listed uncertainties. First of all, I present methods of improving the UAV position accuracy by referencing the position of an UGV using multiple GPS sensors with WCL method and I can improve 1.59 times of positioning accuracy of the UAV. Secondly, I show a method of improving UAV orientation sensing using TPT by integrating a magnetic compass with a solar compass which composed of nine light sensors. Thirdly, dynamic battery consumption rate and part of UAV autonomy are able to be managed by developing the wireless power transformable unmanned ground station using four pairs of transmitter and receiver magnetic loops which has efficiency of 7.85%. Fourthly, error propagation of the mission planning hierarchy is studied and I proved that buffer zone should be $17.84m$ to be safe enough from the error propagation.

In the scalability section, path planning of multiple UAVs, total 18 UAVs, for the surveillance mission is completed by using GA based mTSP and using the method of dividing mTSP into m TSPs with two region division methods such as URD and KVRD. Then, trajectories of heterogeneous UAVs are generated using MUAVS which is run with MATLAB/SIMULINK® and a mission of surveiling a given region within the minimum surveillance time is solved. Oftentimes, the operation time of surveillance becomes the main issue due to limited amount of fuel, so the maximum fuel efficiency is also counted as a counterweight to the minimum surveillance time in a separate chapter. Overall results obtained from previous chapters are used for the cattle roundup application by using feedback linearization controller design.

Future work will focus on finalizing various opened up problems of improving position sensor, orientation sensor, enhancing health management of cooperative UAV operations, and demonstrating robust experimental flight with the analyzed buffer zone size from error propagation. In addition to the future works mentioned at the end of each chapter, game theoretic tracking will be studied by continuing the cattle roundup. I am going to replace two animals with multiple enemy UAVs so that chasing and hunting using multiple UAVs will be studied. Also, methods of reducing personnel per UAV will be studied. According to the Defense Science Study Board, about 17% of UAV accidents (specifically, take-off and landing operations occupy most of the accidents) are resulted from human operator errors [159] and it leads to the necessity of autonomous system for UAVs. Take-off and landing operations are the most critical maneuvers since abrupt acceleration and altitude sensor data changes occur at that moments accompanying relatively large sensor errors. Regarding these problems, there have been many researches on automatic take-off and landing maneuvers of UAVs; three fuzzy logic modules were developed for the autonomous landing control system to control the altitude, the speed, and the position against the runaway [160]; a visual control strategy using optic flow control called *optiPilot* was developed to cope with take-off and landing [161]. Improved sensing and imaging data can bring increased UAV autonomy [4] and I propose that improving data can be partly accomplished with increased number of sensing and imaging data especially when UAV takes off and lands (Reducing Personnel per Asset via Sensing). I will study how to achieve exact altitude, speed, and acceleration data during taking-off and landing by using sensor fusion method.

LIST OF REFERENCES

LIST OF REFERENCES

[1] University of Pennsylvania. Quads Play James Bond: The Making of University of Pennsylvania. https://www.youtube.com/watch?v=OfxS6QBheGo, November 2012. [Online; accessed Dec-01-2012].

[2] FPSRussia. Prototype Quadrotor with Machine Gun. https://www.youtube.com/watch?v=SNPJMk2fgJU, April 2012. [Online; accessed Dec-01-2012].

[3] The Free Dictionary. http://www.thefreedictionary.com/Unmanned+Aerial+Vehicl, December 2012. [Online; accessed Dec-01-2012].

[4] A. Finn and S. Scheding. *Developments and Challenges for Autonomous Unmanned Vehicles: A Compendium*, volume 3. Springer, 1st edition, 2010.

[5] R. Naughton. *Remote Piloted Aerial Vehicles*. Monash University, Februrary 2003.

[6] J. Vujic, A. Marincic, M. Ercegovac, and B. Milovanovic. Nikola Tesla: 145 Years of Visionary Ideas. In *The 5th International Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Service*, volume 1, pages 323–326. IEEE, 2001.

[7] J. W. R. Taylor. *Jane's Pocket Book of Remotely Piloted Vehicles*. Collier Books, 1977.

[8] L. Pearson. Developing the Flying Bomb. *Naval Aviation in World War I*, pages 70–73, 1969.

[9] D. G. Cornelisse. *Splendid Vision, Unswerving Purpose: Developing Air Power for the United States Air Force During the First Century of Powered Flight*. Department of the Air Force, 2003.

[10] D. S. Fahrney. The Birth of Guided Missiles. In *Proceedings of the United States Naval Institute*, pages 54–60, 1980.

[11] G. Goebel. Unmanned Aerial Vehicles: USA. http://www.vectorsite.net/twuav_02.html, February 2012. [Online; accessed Dec-01-2012].

[12] L. Gugerty, D. DeBoom, R. Walker, and J. Burns. Developing a Simulated Uninhabited Aerial Vehicle (UAV) Task Based on Cognitive Task Analysis: Task Analysis Results and Preliminary Simulator Performance Data. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 43, pages 86–90. SAGE Publications, 1999.

[13] E. Bone and C. Bolkcom. Unmanned Aerial Vehicles: Background and Issues for Congress. DTIC Document, 2003.

[14] J. V. McCoy. Unmanned Aerial Logistics Vehicles: A Concept Worth Pursuing. Technical report, DTIC Document, 2003.

[15] Z. Sarris and STN ATLAS. Survey of UAV Applications in Civil Markets (June 2001). In *The 9th Mediterranean Conference on Control and Automation*. IEEE, 2001.

[16] J. Allen and B. Walsh. Enhanced Oil Spill Surveillance, Detection and Monitoring Through the Applied Technology of Unmanned Air Systems. In *International Oil Spill Conference*, 2008.

[17] D. Lupea, I. Szőke, A. Majdik, and G. Lazea. Prototype of a Multi-Robot System for Autonomous Gas Mapping in Polluted Environments. *Hungarian Journal of Industrial Chemistry, Veszprem*, 39(2):237–242, 2011.

[18] J. A. Barnard. The Use of Unmanned Aircraft in Oil, Gas and Mineral E&P Activities. In *Society of Exploration Geophysicists Annual Meeting*, 2008.

[19] N. Kuntz and P. Oh. Development of Autonomous Cargo Transport for an Unmanned Aerial Vehicle using Visual Servoing. In *Proceedings of the Digital Systems and Control Conference*, 2008.

[20] K. Nonami. Prospect and Recent Research & Development for Civil Use Autonomous Unmanned Aircraft as UAV and MAV. *Journal of System Design and Dynamics, J-STAGE*, 1(2):120–128, 2007.

[21] P. Doherty and P. Rudol. A UAV Search and Rescue Scenario with Human Body Detection and Geolocalization. *The 20th Australian Joint Conference on Artificial Intelligence, Springer*, pages 1–13, 2007.

[22] J. McGivering. Drones to Protect Nepal's Endangered Species from Poachers. `http://www.bbc.co.uk/news/science-environment-18527119`, June 2012. [Online; accessed Dec-01-2012].

[23] T. Shima and S. Rasmussen. *UAV Cooperative Decision and Control: Challenges and Practical Approaches*, volume 18. Society for Industrial Mathematics, 2009.

[24] M. E. Dempsey. Eyes of the Army–US Army Roadmap for Unmanned Aircraft Systems 2010–2035. *US Army UAS Center of Excellence, Ft. Rucker, Alabama*, 9, 2010.

[25] K. Dalamagkidis, K. P. Valavanis, and L. A. Piegl. *On Integrating Unmanned Aircraft Systems into the National Airspace System: Issues, Challenges, Operational Restrictions, Certification, and Recommendations*, volume 54. Springer, 2011.

[26] J. Foust and A. S. Boyle. The Strategic Context of Lethal Drones: A Framework for Discussion. *American Security Project*, pages 1–12, August 2012.

[27] N. B. Sarter, R. J. Mumaw, and C. D. Wickens. Pilots' Monitoring Strategies and Performance on Automated Flight Decks: An Empirical Study Combining Behavioral and Eye-Tracking Data. *Human Factors: The Journal of the Human Factors and Ergonomics Society, SAGE Publications*, 49(3):347–357, 2007.

[28] S. D. Manning, C. E. Rash, P. A. Leduc, R. K. Noback, and J. Mckeon. The Role of Human Causal Factors in U.S. Army Unmanned Aerial Vehicle Accidents. Technical report, U.S. Army Medical Research and Materiel Command, March 2004.

[29] J. Uhrmann and A. Schulte. Concept, Design and Evaluation of Congnitive Task-based UAV Guidance. *International Journal on Advances in Intelligent Systems*, 5(1-2):145–158, 2012.

[30] M. L. Cummings, S. Bruni, S. Mercier, and P. J. Mitchell. Automation Architecture for Single Operator, Multiple UAV Command and Control. Technical report, DTIC Document, 2007.

[31] A. G. Shem, T. A. Mazzuchi, and S. Sarkani. Addressing Uncertainty in UAV Navigation Decision-Making. *Transactions on Aerospace and Electronic Systems, IEEE*, 44(1):295–313, 2008.

[32] H. Durrant-Whyte, M. Stevens, and E. Nettleton. Data Fusion in Decentralised Sensing Networks. In *The 4th International Conference on Information Fusion*, pages 302–307, 2001.

[33] Z. Wu, H. Kumar, and A. Davari. Performance Evaluation of OFDM Transmission in UAV Wireless Communication. In *Proceedings of the 37th Southeastern Symposium on System Theory*, pages 6–10. IEEE, 2005.

[34] J. Tisdale, Z. Kim, and J. Hedrick. Autonomous UAV Path Planning and Estimation. *Robotics & Automation Magazine, IEEE*, 16(2):35–42, 2009.

[35] J. Bellingham. *Multi-Task Allocation and Path Planning for Cooperating UAVs*. PhD thesis, Citeseer, 2001.

[36] S. A. Bortoff. Path Planning for UAVs. In *Proceedings of the American Control Conference*, volume 1, pages 364–368. IEEE, 2000.

[37] F.L. Lian and R. Murray. Real-Time Trajectory Generation for the Cooperative Path Planning of Multi-Vehicle Systems. In *Proceedings of the 41st Conference on the Decision and Control*, volume 4, pages 3766–3769. IEEE, 2002.

[38] L. Singh and J. Fuller. Trajectory Generation for a UAV in Urban Terrain, Using Nonlinear MPC. In *Proceedings of the American Control Conference*, volume 3, pages 2301–2308. IEEE, 2001.

[39] M. Farhood and E. Feron. Obstacle-Sensitive Trajectory Regulation via Gain Scheduling and Semidefinite Programming. *Transactions on Control Systems Technology, IEEE*, 20(4):1107–1115, 2012.

[40] P. B. Sujit and R. Beard. Distributed Sequential Auctions for Multiple UAV Task Allocation. In *American Control Conference*, pages 3955–3960. IEEE, 2007.

[41] P. Sujit, A. Sinha, and D. Ghose. Team, Game, and Negotiation Based Intelligent Autonomous UAV Task Allocation for Wide Area Applications. *Innovations in Intelligent Machines-1, Springer*, pages 39–75, 2007.

[42] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The Grasp Multiple Micro-UAV Testbed. *Robotics & Automation Magazine, IEEE*, 17(3):56–65, 2010.

[43] B. Bethke, M. Valenti, and J. How. Cooperative Vision Based Estimation and Tracking using Multiple UAVs. In *Advances in Cooperative Control and Optimization*, pages 179–189. Springer, 2007.

[44] S. Leven, J. Zufferey, and D. Floreano. Dealing with Midair Collisions in Dense Collective Aerial Systems. *Journal of Field Robotics, Wiley Online Library*, 28(3):405–423, 2011.

[45] A. Ryan, J. Tisdale, M. Godwin, D. Coatta, D. Nguyen, S. Spry, R. Sengupta, and J. K. Hedrick. Decentralized Control of Unmanned Aerial Vehicle Collaborative Sensing Missions. In *American Control Conference*, pages 4672–4677. IEEE, 2007.

[46] M. Barczyk and A. F. Lynch. Integration of a Triaxial Magnetometer into a Helicopter UAV GPS-Aided INS. *Transactions on Aerospace and Electronic Systems, IEEE*, 48(4):2947–2960, 2012.

[47] A. Nemra and N. Aouf. Robust INS/GPS Sensor Fusion for UAV Localization using SDRE Nonlinear Filtering. *Sensors Journal, IEEE*, 10(4):789–798, 2010.

[48] J. Ryu and J. C. Gerdes. Integrating Inertial Sensors with GPS for Vehicle Dynamics Control. *Journal of Dynamic Systems, Measurement, and Control, ASME*, 126(2):243–254, 2004.

[49] H. Qi and J. B. Moore. Direct Kalman Filtering Approach for GPS/INS Integration. *Transactions on Aerospace and Electronic Systems, IEEE*, 38(2):687–693, 2002.

[50] D. Gebre-Egziabher, J. D. Powell, and P. Enge. Design and Performance Analysis of a Low-Cost Aided-Dead Reckoning Navigation System. *Gyroscopy and Navigation*, 4(35):83–92, 2001.

[51] D. Lee, Y. Kim, and H. Bang. Vision-based Terrain Referenced Navigation for Unmanned Aerial Vehicles using Homography Relationship. *Journal of Intelligent and Robotic Systems, Springer*, pages 1–9, 2012.

[52] A. Cesetti, E. Frontoni, A. Mancini, P. Zingaretti, and S. Longhi. A Vision-based Guidance System for UAV Navigation and Safe Landing Using Natural Landmarks. *Journal of Intelligent and Robotic Systems, Springer*, 57(1):233–257, 2010.

[53] D. K. Schrader, B. C. Min, E. T. Matson, and J. E. Dietz. Combining Multiple, Inexpensive GPS Receivers to Improve Accuracy and Reliability. In *Sensors Applications Symposium*, pages 1–6. IEEE, 2012.

[54] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less Low-Cost Outdoor Localization for Very Small Devices. *Personal Communications, IEEE*, 7(5):28–34, 2000.

[55] J. Blumenthal, R. Grossmann, F. Golatowski, and D. Timmermann. Weighted Centroid Localization in Zigbee-based Sensor Networks. In *International Symposium on Intelligent Signal Processing*, pages 1–6. IEEE, 2007.

[56] R. G. Brown and P. Y. C. Hwang. *Introduction to Random Singals and Applied Kalman Filtering*. Wiley, 3rd edition, 1997.

[57] T. J. Nugent and J. T. Kare. Laser Power for UAVs. *LaserMotive, White Paper*, 2010.

[58] T. J. Nugent and J. T. Kare. Laser Power Beaming for Defense and Security Applications. In *Proceedings of Society of Photographic Instrumentation Engineers*, volume 8045, pages 804514–1–804514–8, 2011.

[59] T. J. Nugent, J. T. Kare, D. Bashford, C. Erickson, and J. Alexander. 12-Hour Hover: Flight Demonstration of a Laser-Powered Quadrocopter. Technical report, 2011.

[60] NyARToolkit. NyARToolkit Project. `http://nyatla.jp/nyartoolkit/wp/?page\_id=198`, February 2012. [Online; accessed Dec-27-2012].

[61] D. B. Kingston and R. W. Beard. Real-time Attitude and Position Estimation for Small UAVs Using Low-cost Sensors. In *The 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, pages 2004–6488. Citeseer, 2004.

[62] S. Jung and K. B. Ariyur. Enabling Operational Autonomy for UAVs with Scalability. *Journal of Aerospace Computing, Information, and Communications (accepted on Feb 26, 2013 and waiting for publication), AIAA*, 2013.

[63] SÖ. Akdemir and E. Öztekin. Generating Function for Rotation Matrix Elements. *International Journal of Quantum Chemistry, Wiley Online Library*, 112(2):367–372, 2012.

[64] S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and Control of an Indoor Micro Quadrotor. In *Proceedings of the International Conference on Robotics and Automation*, volume 5, pages 4393–4398. IEEE, 2004.

[65] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB(Springer Tracks in Advanced Robotics)*. Springer, 2nd edition, March 2013.

[66] P. E. I. Pounds. *Design, Construction and Control of a Large Quadrotor Micro Air Vehicle*. PhD thesis, The Australian National University, September 2007.

[67] B. Dutton and T. J. Cutler. *Dutton's Nautical Navigation*. Naval Institute Press, 2004.

[68] J. Evans. *The History and Practice of Ancient Astronomy*. Oxford University Press, USA, 1998.

[69] I. Reda and A. Andreas. Solar Position Algorithm for Solar Radiation Applications. Technical report, National Renewable Energy Laboratory, January 2008.

[70] I. Reda and A. Andreas. Solar Position Algorithm for Solar Radiation Applications. *Solar energy, Elsevier*, 76(5):577–589, 2004.

[71] J. Barnes and K. B. Ariyur. Miniaturizing the Spherical Sundial: A Hemispherical Sensor for Orientation and Positioning with Respect to Point Sources of Light. In *International Conference on Control Applications*, pages 662–667. IEEE, 2011.

[72] K. A. O. Suzuki, P. Kemper Filho, and J. R. Morrison. Automatic Battery Replacement System for UAVs: Analysis and Design. *Journal of Intelligent & Robotic Systems, Springer*, 65(1):563–586, 2012.

[73] K. A. Swieringa, C. B. Hanson, J. R. Richardson, J. D. White, Z. Hasan, E. Qian, and A. Girard. Autonomous Battery Swapping System for Small-Scale Helicopters. In *International Conference on Robotics and Automation*, pages 3335–3340. IEEE, 2010.

[74] F. P. Kemper, K. A. O. Suzuki, and J. R. Morrison. UAV Consumable Replenishment: Design Concepts for Automated Service Stations. *Journal of Intelligent & Robotic Systems, Springer*, 61(1):369–397, 2011.

[75] M. Valenti, D. Dale, J. P. How, D. P. De Farias, and J. Vian. Mission Health Management for 24/7 Persistent Surveillance Operations. In *Proceedings of the Guidance, Navigation, and Control Conference*. AIAA, August 2007.

[76] W. C. Brown and E. E. Eves. Beamed Microwave Power Transmission and Its Application to Space. *Transactions on Microwave Theory and Techniques, IEEE*, 40(6):1239–1250, 1992.

[77] W. C. Brown. Experimental Airborne Microwave Supported Platform. Technical report, Defense Technical Information Center Document, 1965.

[78] E. J. Silberg and J. H. Milgram. Battery Charging Arrangement for Unmanned Aerial Vehicle Utilizing the Electromagnetic Field Associated with Utility Power Lines to Generate Power to Inductively Charge Energy Supplies, May 2010. US Patent 7,714,536.

[79] B. Griffin and C. Detweiler. Resonant Wireless Power Transfer to Ground Sensors from a UAV. In *International Conference on Robotics and Automation*, pages 2660–2665. IEEE, 2012.

[80] R. S. Elliot. *Electromagnetics: History, Theory, and Applications*. IEEE, 1st edition, 1993.

[81] R. P. Redwine and J. Conrad. *Inductance and Magnetic Energy*. MIT, September 2012.

[82] Digi Key. IRF510. `http://www.digikey.com/product-detail/en/IRF510PBF/IRF510PBF-ND/811710`, September 2012. [Online; accessed Dec-27-2012].

[83] T. Bresciani. *Modelling, Identification and Control of a Quadrotor Helicopter*. Master's thesis, Department of Automatic Control, Lund University, Sweden, October 2008.

[84] H. T. Friis. A Note on a Simple Transmission Formula. *Proceeding of the Institute of Radio Engineers and Waves and Electrons*, 34(5):254–256, 1946.

[85] D. Mascareñas, E. Flynn, M. Todd, G. Park, and C. Farrar. Wireless Sensor Technologies for Monitoring Civil Structures. *Sound and Vibration, Bay Village, Ohio, Acoustical Publications, inc.*, 42(4):16–21, 2008.

[86] Parrot SA. AR.Drone. `http://ardrone2.parrot.com/`, February 2013. [Online; accessed Mar-14-2013].

[87] J. F. Smith III and T. H. Nguyen. Fuzzy Logic Based UAV Allocation and Coordination. In *Informatics in Control Automation and Robotics*, pages 81–94. Springer, 2008.

[88] N. Shorter and T. Kasparis. Automatic Vegetation Identification and Building Detection from a Single Nadir Aerial Image. *Remote Sensing, Molecular Diversity Preservation International*, 1(4):731–757, 2009.

[89] I. Lee, C. T. Bruin, and K. Lee. Map Segmentation for Geospatial Data Mining Through Generalized Higher-Order Voronoi Diagrams with Sequential Scan Algorithms. *Expert Systems with Applications, Elsevier*, 39(12):11135–11148, September 2012.

[90] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*, volume 501. Wiley, 2nd edition, 2009.

[91] P. Bhattacharya and M. L. Gavrilova. Roadmap-Based Path Planning-Using the Voronoi Diagram for a Clearance-Based Shortest Path. *Robotics & Automation Magazine, IEEE*, 15(2):58–66, 2008.

[92] M. Šeda and V. Pich. Robot Motion Planning Using Generalised Voronoi Diagrams. *The 8th WSEAS International Conference on Signal Processing, Computational Geometry and Artificial Vision*, 1:215–220, 2008.

[93] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin. Path Planning for Mobile Robot Navigation Using Voronoi Diagram and Fast Marching. In *International Conference on Intelligent Robots and Systems, IEEE*, pages 2376–2381. IEEE, 2006.

[94] F. Aurenhammer. Voronoi Diagrams-A Survey of a Fundamental Geometric Data Structure. *Computing Surveys (CSUR), ACM*, 23(3):345–405, 1991.

[95] C. K. Yap. An O($n$log$n$) Algorithm for the Voronoi Diagram of a Set of Simple Curve Segments. *Discrete & Computational Geometry, Springer*, 2(1):365–393, 1987.

[96] E. W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische mathematik, Springer*, 1(1):269–271, 1959.

[97] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan. Fuzzy Dijkstra Algorithm for Shortest Path Problem under Uncertain Environment. *Applied Soft Computing, Elsevier*, 12(3):1231–1237, 2012.

[98] A. Király and J. Abonyi. Optimization of Multiple Traveling Salesmen Problem by a Novel Representation Based Genetic Algorithm. *Intelligent Computational Optimization in Engineering, Springer*, pages 241–269, 2011.

[99] T. Bektas. The Multiple Traveling Salesman Problem: An Overview of Formulations and Solution Procedures. *Omega, Elsevier*, 34(3):209–219, 2006.

[100] A. Valero-Gomez, J. Valero-Gomez, A. Castro-Gonzalez, and L. Moreno. Use of Genetic Algorithms for Target Distribution and Sequencing in Multiple Robot Operations. pages 2718–2724. Proceedings of the International Conference on Robotics and Biomimetics, December 2011.

[101] E. Edison and T. Shima. Integrated Task Assignment and Path Optimization for Cooperating Uninhabited Aerial Vehicles using Genetic Algorithms. *Computers & Operations Research, Elsevier*, 38(1):340–356, 2011.

[102] F. C. J. Allaire, M. Tarbouchi, G. Labonté, and G. Fusina. FPGA Implementation of Genetic Algorithm for UAV Real-Time Path Planning. In *Unmanned Aircraft Systems*, pages 495–510. Springer, 2009.

[103] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 3rd edition, 2008. pages 279–292.

[104] S. Jackson, J. Tisdale, M. Kamgarpour, B. Basso, and J. K. Hedrick. Tracking Controllers for Small UAVs with Wind Disturbances: Theory and Flight Results. In *The 47th Conference on Decision and Control*, pages 564–569. IEEE, 2008.

[105] J. P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-Time Indoor Autonomous Vehicle Test Environment. *Control Systems, IEEE*, 28(2):51–64, 2008.

[106] B. Bethke, J. P. How, and J. Vian. Group Health Management of UAV Teams with Applications to Persistent Surveillance. In *American Control Conference*, pages 3145–3150. IEEE, 2008.

[107] B. Bethke, M. Valenti, and J. P. How. UAV Task Assignment. *Robotics & Automation Magazine, IEEE*, 15(1):39–44, 2008.

[108] M. Valenti, B. Bethke, J. P. How, D. P. de Farias, and J. Vian. Embedding Health Management into Mission Tasking for UAV Teams. In *American Control Conference*, pages 5777–5783. IEEE, 2007.

[109] F. Taylor. New Google Earth Satellite Launching Today. `http://www.gearthblog.com/blog/archives/2007/09/new_google_earth_satellite_launchin.html`, September 2007. [Online; accessed Dec-18-2012].

[110] A. Chang, M. Parrales, J. Jimenez, M. Sobieszczyk, S. Hammer, D. Copenhaver, and R. Kulkarni. Combining Google Earth and GIS Mapping Technologies in a Dengue Surveillance System for Developing Countries. *International Journal of Health Geographics, BioMed Central Ltd*, 8(1):49, 2009.

[111] D. Potere. Horizontal Positional Accuracy of Google Earths High-Resolution Imagery Archive. *Sensors, Molecular Diversity Preservation International*, 8(12):7973–7981, 2008.

[112] K. W. Hall, J. K. Cooper, and D. C. Lawton. GPS Accuracy: Hand-held Versus RTK. *CREWES Research Report*, 20, 2008.

[113] SOKKIA. GSR2700 ISX: Operations Manual. http://www.sistemastopograficos.com.mx/manuales/manuales_gps/GSR%202700%20ISX%20-%20Operations%20Manual.pdf, Feb 2007. [Online; accessed Jun-26-2013].

[114] D. Grigillo, M. K. Fras, and D. Petrovič. Automated Building Extraction from IKONOS Images in Suburban Areas. *International Journal of Remote Sensing, Taylor & Francis*, 33(16):5149–5170, 2012.

[115] X. C. He and N. H. C. Yung. Corner Detector Based on Global and Local Curvature Properties. *Optical Engineering, International Society for Optics and Photonics*, 47(5):057008–1–057008–12, 2008.

[116] K. E. Hoff III, J. Keyser, M. Lin, D. Manocha, and T. Culver. Fast Computation of Generalized Voronoi Diagrams using Graphics Hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 277–286. ACM Press/Addison-Wesley Publishing Co., 1999.

[117] K. B. Ariyur and K. O. Fregene. Autonomous Tracking of a Ground Vehicle by a UAV. In *American Control Conference*, pages 669–671. IEEE, 2008.

[118] Scout UAV Store. MediaTek MT3329 GPS. http://store.scoutuav.com/product/add-ons/mediatek-mt3329-gps-10hz-adapter-basic/, June 2013. [Online; accessed Jun-26-2013].

[119] 3D Robotics. MPXV7002DP Airspeed Kit. http://store.3drobotics.com/products/airspeed-kit-with-mpxv7002dp, June 2013. [Online; accessed Jun-26-2013].

[120] NitroPlanes. Remote Control RC UAV Drone Airplane. http://www.nitroplanes.com/projet-drone-2500mm-kit.html, February 2013. [Online; accessed Mar-14-2013].

[121] D.o.t.A. Headquarters. "Eyes of the Army" U.S. Army Roadmap for Unmanned Aircraft Systems 2010-2035. *U.S. Army UAS Center of Excellence (ATZQ-CDI-C)*, 2010.

[122] S. Mittal and K. Deb. Three-Dimensional Offline Path Planning for UAVs Using Multiobjective Evolutionary Algorithms. In *Proceedings of the Congress on Evolutionary Computation*, pages 3195–3202, 2007.

[123] R. W. Beard, T. W. McLain, M. A. Goodrich, and E. P. Anderson. Coordinated Target Assignment and Intercept for Unmanned Air Vehicles. *Transactions on Robotics and Automation, IEEE*, 18(6):911–922, 2002.

[124] T. W. McLain, P. R. Chandler, S. Rasmussen, and M. Pachter. Cooperative Control of UAV Rendezvous. In *Proceedings of the IEEE American Control Conference*, volume 3, pages 2309–2314, 2001.

[125] P. R. Chandler, M. Pachter, and S. Rasmussen. UAV Cooperative Control. In *Proceedings of the American Control Conference*, volume 1, pages 50–55. IEEE, 2001.

[126] T. W. McLain and R. W. Beard. Trajectory Planning for Coordinated Rendezvous of Unmanned Air Vehicles. In *Proceedings of the Guidance, Navigation, and Control Conference*, volume 4369, pages 1–8, 2000.

[127] M. Šeda and V. Pich. Robot Motion Planning Using Generalised Voronoi Diagrams. *The 8th International Conference on Signal Processing, Computational Geometry and Artificial Vision, WSEAS*, 1:215–220, 2008.

[128] X. Liu, T. Kanungo, and R. M. Haralick. On the Use of Error Propagation for Statistical Validation of Computer Vision Software. *Transactions on Pattern Analysis and Machine Intelligence, IEEE*, 27(10):1603–1614, 2005.

[129] J. C. Doyle, B. A. Francis, and A. Tannenbaum. *Feedback Control Theory*, volume 1. Macmillan, New York, 1992.

[130] J. T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics, AIAA*, 21(2), 2012.

[131] M. Kamgarpour, V. Dadok, and C. Tomlin. Trajectory Generation for Aircraft Subject to Dynamic Weather Uncertainty. In *The 49th Conference on Decision and Control*, pages 2063–2068. IEEE, 2010.

[132] C. G. Carlson and D. E. Clay. The Earth Model–Calculating Field Size and Distances Between Points Using GPS Coordinates. *Site Specific Management Guidelines, Potash & Phosphate Institute (PPI)*, pages 1–4, 1999.

[133] H. Spath. *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*. Ellis Horwood, Ltd., 1980.

[134] M. L. Fredman and R. E. Tarjan. Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the Association for Computing Machinery, ACM*, 34(3):596–615, 1987.

[135] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Optimization. *Transactions on Evolutionary Computation, IEEE*, 1(1):67–82, 1997.

[136] M. Bellmore and S. Hong. Transformation of Multisalesman Problem to the Standard Traveling Salesman Problem. *Journal of the Association for Computing Machinery, ACM*, 21(3):500–504, 1974.

[137] S. Hong and M. W. Padberg. A Note on the Symmetric Multiple Traveling Salesman Problem with Fixed Charges. *Operations Research, JSTOR*, pages 871–874, 1977.

[138] S. S. Skiena. *The Algorithm Design Manual*, volume 1. Springer, 1998.

[139] W. Barnes and W. McCormick. *Aerodynamics, Aeronautics, and Flight Mechanics*. 2nd edition, September 1995.

[140] Y. J. Zhao and Y. C. Qi. Minimum Fuel Powered Dynamic Soaring of Unmanned Aerial Vehicles Utilizing Wind Gradients. *Optimal Control Applications and Methods, Wiley Online Library*, 25(5):211–233, 2004.

[141] J. Foo, J. Knutzon, J. Oliver, and E. Winer. Three-Dimensional Path Planning of Unmanned Aerial Vehicles using Particle Swarm Optimization. In *The 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Portsmouth, Virginia*, 2006.

[142] S. Jung and K. B. Ariyur. Scalable Autonomy for UAVs. pages 1–16. AIAA Infotech@Aerospace, St. Louis, MO, March 2011.

[143] R. Vidal, O. Shakernia, H. J. Kim, D. H. Shim, and S. Sastry. Probabilistic Pursuit-Evasion Games: Theory, Implementation, and Experimental Evaluation. *Transactions on Robotics and Automation, IEEE*, 18(5):662–669, 2002.

[144] U. Zengin and A. Dogan. Real-Time Target Tracking for Autonomous UAVs in Adversarial Environments: A Gradient Search Algorithm. *Transactions on Robotics, IEEE*, 23(2):294–307, 2007.

[145] S. D. Bopardikar, F. Bullo, and J. Hespanha. A Cooperative Homicidal Chauffeur Game. In *The 46th Conference on Decision and Control*, pages 4857–4862. IEEE, 2007.

[146] P. Fabiani, H. H. González-Baños, J. C. Latombe, and D. Lin. Tracking an Unpredictable Target among Occluding Obstacles under Localization Uncertainties. *Robotics and Autonomous Systems, Elsevier*, 38(1):31–48, 2002.

[147] A. Das, F. Lewis, and K. Subbarao. Backstepping Approach for Controlling a Quadrotor using Lagrange Form Dynamics. *Journal of Intelligent and Robotic Systems, Springer*, 56(1-2):127–151, 2009.

[148] T. Madani and A. Benallegue. Backstepping Control for a Quadrotor Helicopter. In *International Conference on Intelligent Robots and Systems*, pages 3255–3260. IEEE, 2006.

[149] T. Madani and A. Benallegue. Control of a Quadrotor Mini-Helicopter via Full State Backstepping Technique. In *The 45th Conference on Decision and Control*, pages 1515–1520. IEEE, 2006.

[150] E. Altuğ, J. P. Ostrowski, and C. J. Taylor. Control of a Quadrotor Helicopter using Dual Camera Visual Feedback. *The International Journal of Robotics Research, SAGE Publications*, 24(5):329–341, 2005.

[151] S. Bouabdallah and R. Siegwart. Full Control of a Quadrotor. In *International Conference on Intelligent Robots and Systems*, pages 153–158. IEEE, 2007.

[152] D. Lee, H. Kim, and S. Sastry. Feedback Linearization vs. Adaptive Sliding Mode Control for a Quadrotor Helicopter. *International Journal of Control, Automation and Systems, Springer*, 7(3):419–428, 2009.

[153] A. Mokhtari, A. Benallegue, and B. Daachi. Robust Feedback Linearization and GH_∞ Controller for a Quadrotor Unmanned Aerial Vehicle. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1198–1203. IEEE, 2005.

[154] A. Benallegue, A. Mokhtari, and L. Fridman. Feedback Linearization and High Order Sliding Mode Observer for a Quadrotor UAV. In *International Workshop on Variable Structure Systems*, pages 365–372. IEEE, 2006.

[155] H. Voos. Nonlinear Control of a Quadrotor Micro-UAV using Feedback-Linearization. In *International Conference on Mechatronics*, pages 1–6. IEEE, 2009.

[156] B. Erginer and E. Altug. Modeling and PD Control of a Quadrotor VTOL Vehicle. In *Intelligent Vehicles Symposium*, pages 894–899. IEEE, 2007.

[157] A. Tayebi and S. McGilvray. Attitude Stabilization of a VTOL Quadrotor Aircraft. *Transactions on Control Systems Technology, IEEE*, 14(3):562–571, 2006.

[158] R. Sepulchre, M. Janković, and P. V. Kokotović. *Constructive Nonlinear Control*. Communications and Control Engineering Series. Springer London, Limited, 2012.

[159] Defense Science Study Board. *Unmanned Aerial Vehicles and Uninhabited Combat Aerial Vehicles*. Office of the Under Secretary of Defense for Acquisition, Technology and Logistics, Washington, DC, 20301-3140, 2004.

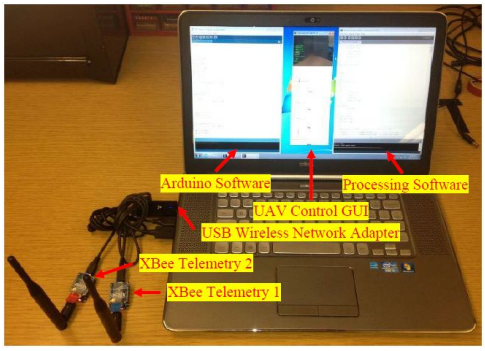
[160] S. Kurnaz and O. Çetin. Autonomous Navigation and Landing Tasks for Fixed Wing Small Unmanned Aerial Vehicles. *Acta Polytechnica Hungarica*, 7(1):87–102, 2010.

[161] A. Beyeler, J. C. Zufferey, and D. Floreano. optiPilot: Control of Take-off and Landing using Optic Flow. In *Proceedings of the European Micro Air Vehicle Conference*, volume 27, pages 201–219, 2009.

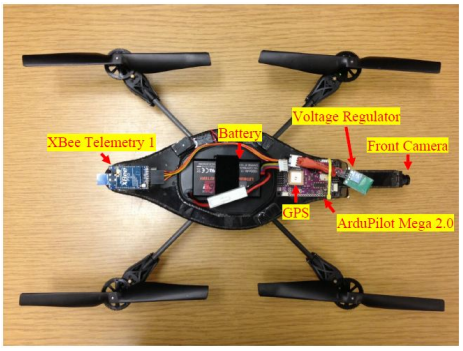
APPENDICES

Appendix A. Experiment Setup

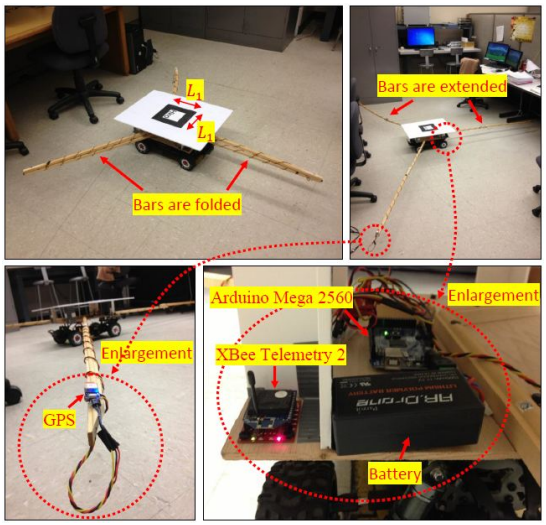
An AR.Drone (20.7x20.3in diameter and 420g weight, Figure A.1(b)) [86] designed by Parrot SA company is used for experiments by integrating an ArduPilot Mega 2.0 (APM) microcontroller. AR.Drone is controlled by Processing software on the Unmanned Ground Station (UGS, Figure A.1(a)) using Wi-Fi (b,g,n) signals and APM can perform two-way communication by Arduino software on the UGS using $2.4GHz$ radio signals (Figure A.1(c)). The AR.Drone System includes an AR.Drone, a MB1200 XL-MaxSonar-EZ0 ultrasonic range finder, a set of XBee telemetry kit, a AnyVolt Micro Universal DC-DC converter, and ArduPilot Mega 2.0 microcontroller which includes a MediaTek MT3329 GPS, 6-axis accelerometer/gyro, 3-axis magnetometer, an ultrasound sensor, and an ultrasound range finder. In addition, AR.Drone itself also contains a 6-axis accelerometer/gyro, 3-axis magnetometer, an ultrasound altimeter, and two cameras (640x480 pixels VGA). One pair of Accelerometers, gyros, and ultrasound sensors are integrated to achieve better data.



(a) UGS system setup.

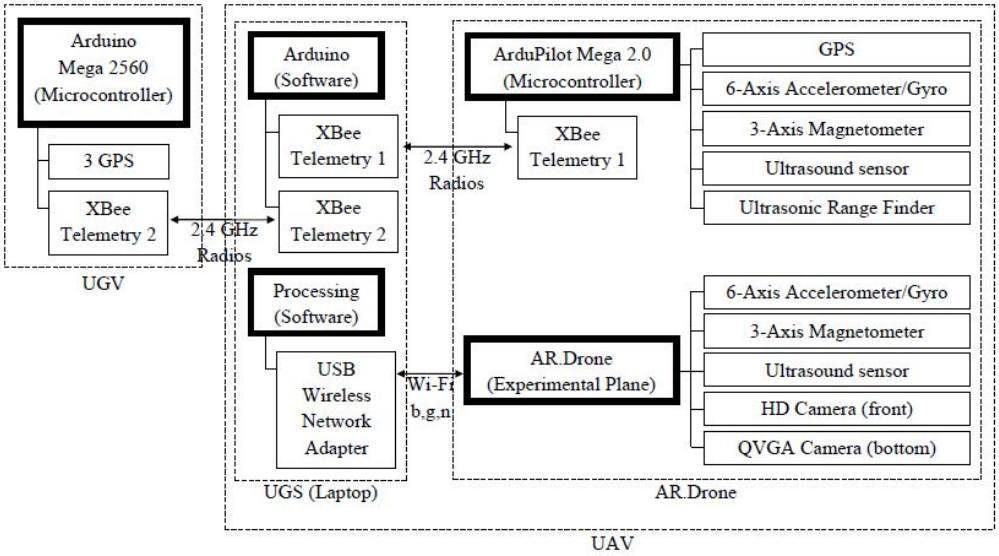


(b) UAV system setup.



(c) UGV system setup.

(d) Outdoor experiment.



(e) Overall communication setup.

Figure A.1.Experiment setup.

Appendix B. TSP with GA

The GA starts with an initial population of n node or edge traverses which are randomly chosen, then evaluates them with the fitness function of time of traverse to calculate the $m \times n$ matrix, $P(0)$, where m is the number of UAVs. If calculated times are not smaller than a constant T_G and the number of iterations is less than the maximum permissible, new populations are created using crossover and mutation (flip, swap, and slide) as shown in Figure B.1.

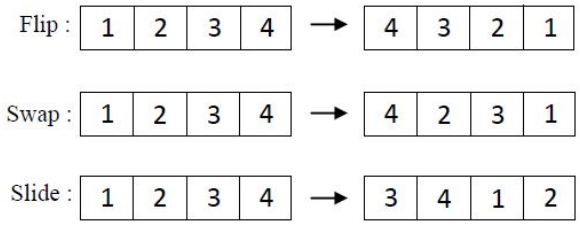
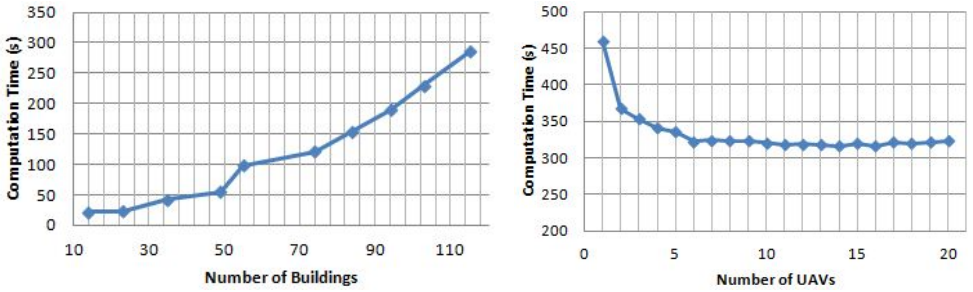


Figure B.1.Flip, swap, and slide operations for mutation.

With a population size P, I choose the maximum number of iterations as $N = \|c\frac{BP}{m}\|$ where $c = 2.5$ is a user-defined amplitude parameter, B is a total number of buildings, and m is a total number of UAVs. The offline computation time for the GA (Algorithm 1) grows quadratically with increasing number of buildings as shown in Figure B.2(a). Also, with given same number of buildings, the GA computation time flat lines with the number of UAVs because of my choice of number of iteration N as shown in Figure B.2(b). TSP with GA keeps iterating with a while loop until the total traveling time becomes less than T_G or until the maximum number of iterations is reached [103]. All search and surveillance problems can be encoded as Algorithm 1, as they involve trips to specific nodes, or edges, or points on edges.



(a) With fixed number of 6 UAVs. (b) With fixed number of 114 buildings.

Figure B.2.GA computation time.

Algorithm 1 GA (for mTSP).

```
for i = 1 : m do
  while min_time > T_G do
    for iter = 1 : num_iter do
      min_time = calculated minimum time from total distance lists;
      if min_time < global_min then
        global_min = min_time;
        GA (flip, swap, slide operations);
      end if
    end for
  end while
end for
```

Appendix C. Block Separation Algorithm (BSA)

I developed the BSA to separate and number the blocks generated by the RGA. The output of the RGA shown in Figure 6.2(b) is a matrix M of all the N points composing blocks in a given satellite photograph (Figure C.1). In the planar case the matrix M has 2 columns. For $i = 1 : N$,

1. Start from point $P_i = \{M_{i,x}, M_{i,y}\}$.
2. Find all points within a distance range $[\underline{l}, \bar{l}]$ from P_i.
3. Mark the set of points as closed if the distance of point P_k to P_i lies in $[\underline{l}, \bar{l}]$.
4. Erase points near the block without neighbors.
5. Repeat steps 1-4 with points not in detected blocks until no more points are available.

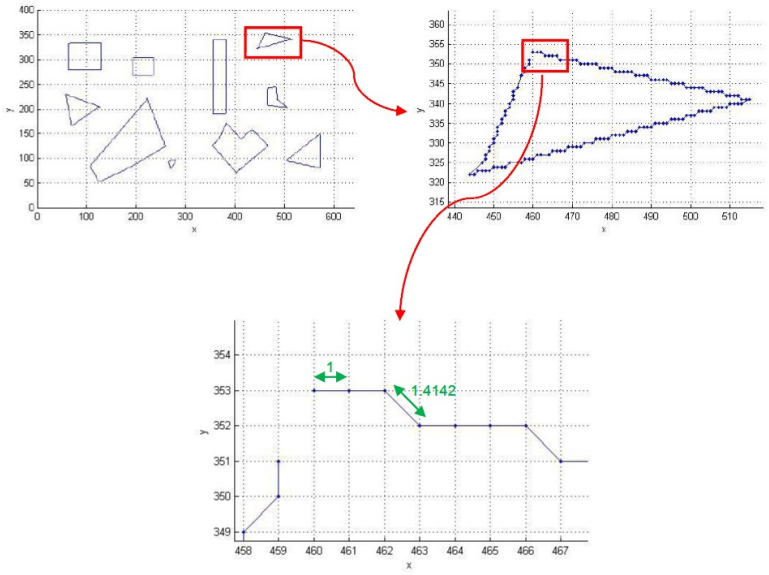


Figure C.1.Block Separation Algorithm (BSA).

Appendix D. Vehicle Properties

All UAVs share same system properties of Table D.1.

Table D.1Shared system properties of UAVs.

Altitude_{max}	$50 \quad m$
Altitude_{normal}	$30 \quad m$
$\text{Acceleration}_{max}$	$10 \quad m/s^2$
$[V_x, V_y, V_z]_{initial}$	$[0, 0, 0] \quad m/s$
Time Step of SIMULINK®	$0.01 \quad s$
The Field of View of Camera	$40°$
τ_x	$0.25 \quad s$
τ_v	$0.5 \quad s$

Table D.2System properties of a fixed-wing UAV.

Velocity_{min}	$5 \quad m/s$
Velocity_{max}	$10 \quad m/s$
Velocity_{max}	$5 \quad m/s$ (for fuel efficiency)
Altitude_{min}	$25 \quad m$

Table D.3System properties of a hover capable UAV.

Velocity_{min}	$0 \quad m/s$
Velocity_{max}	$5 \quad m/s$
Altitude_{min}	$3 \quad m$

Appendix E. MultiUAV System (MUAVS)

E.1 MUAVS Communication Structure

MUAVS used in this paper consists of 1 leader UAV and follower UAVs which can increase as many as researchers want. The leader UAV takes all the information of follower UAVs and used these information to perform some missions, Collision Avoidance System (CAS), etc. The structure of the MUAVS is

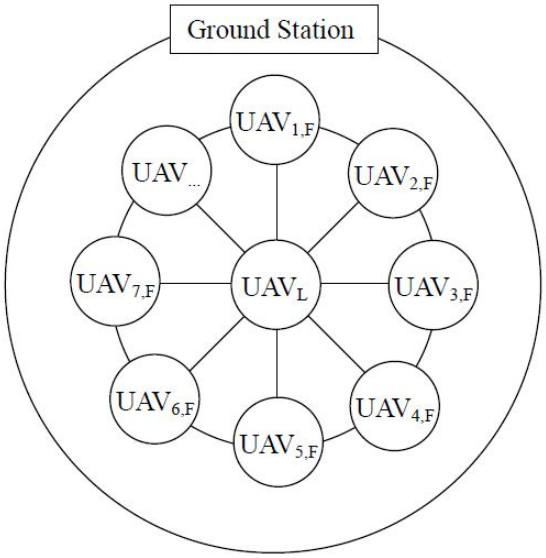


Figure E.1.Structure of MUAVS.

where UAV_L is a leader UAV and $\text{UAV}_{i,F}$ is a follower UAV where $i \in \{1, 2, ..., m-1\}$ and m is the total number of UAVs. Ground station can operate UAVs individually and all $\text{UAV}_{i,F}$ can communicate each other. If UAV_L is crashed during missions, another UAV takes over the leader tasks. The reason for using only one leader UAV is to minimize online computation time. During flights, each UAV can identify targets and land on when it reaches the goal position.

E.2 MUAVS Input Process

To run MUAVS, proper inputs should be plugged in before running MUAVS simulation. There are total 6 user-defined inputs; total number of UAVs, types of UAVs, initial states of UAVs, mission types, target images to search, and a satellite image to surveil.

1. Total number of UAVs can vary as researchers want from 1 to hundreds. However, researchers need to be careful that increasing number of UAVs introduce additional computation time which delays the simulation (default: 18 UAVs).
2. Researchers can variate types of UAVs as homogenous or heterogeneous between fixed-wing UAVs or hover capable UAVs (default: fixed-wing UAVs).
3. Researchers can assign same initial state to all UAVs or assign different initial states individually (default: $P_0 = (0, 0, 0)$ and $v_0 = (0, 0, 0)$).
4. There are 2 mission types; fly from one point to the other point; search all regions (default: fly from one point to the other point).
5. If mission type of searching all region is selected in the previous process, researchers need to select a region division type between URD and KVRD (default: URD).
6. Researchers can choose one option of target handling among three options; disregard, track, and fire (default: disregard).
7. Researchers can input more as many as target images, but need to be careful using many target images since it delays simulation (default: no target image).
8. Researchers can input one satellite image (default: satellite image with 10 buildings).

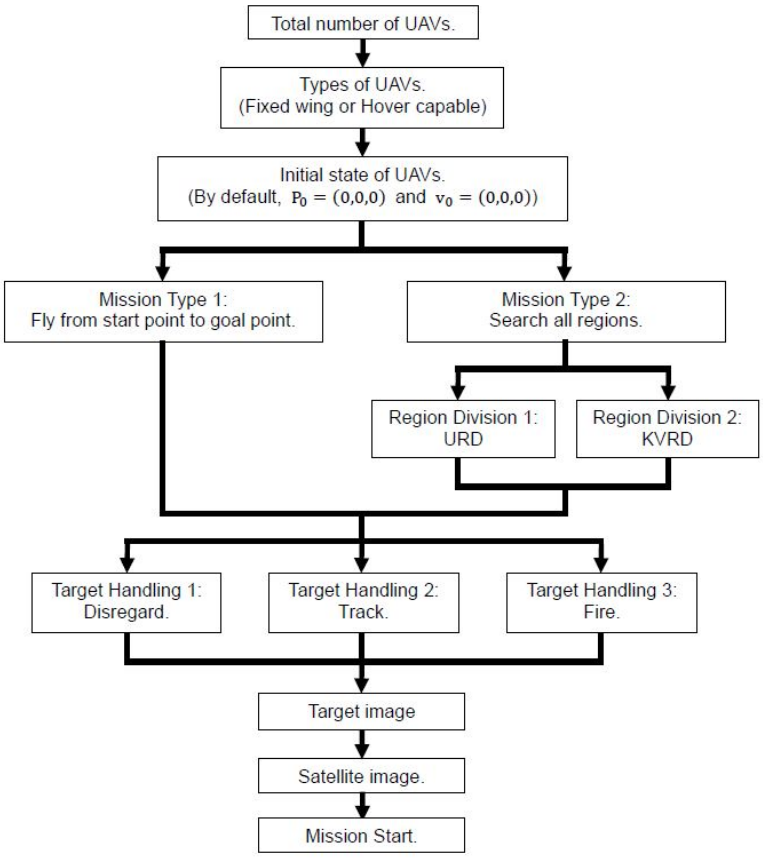


Figure E.2.Process of inserting inputs for MUAVS.

E.3 MUAVS MATLAB/SIMULINK® Organization

UAV trajectories are generated by MATLAB® scripts and dynamics of UAVs are generated by SIMULINK®. Hierarchy of MATLAB® script for generating trajectories are covered in Chapter 2. There are total 3 levels in SIMULINK® models and level 1 is the top-level block and level 3 is the lowest level block.

The number of UAVs can increase as many as researchers want, but it is set as 18 UAVs by default.

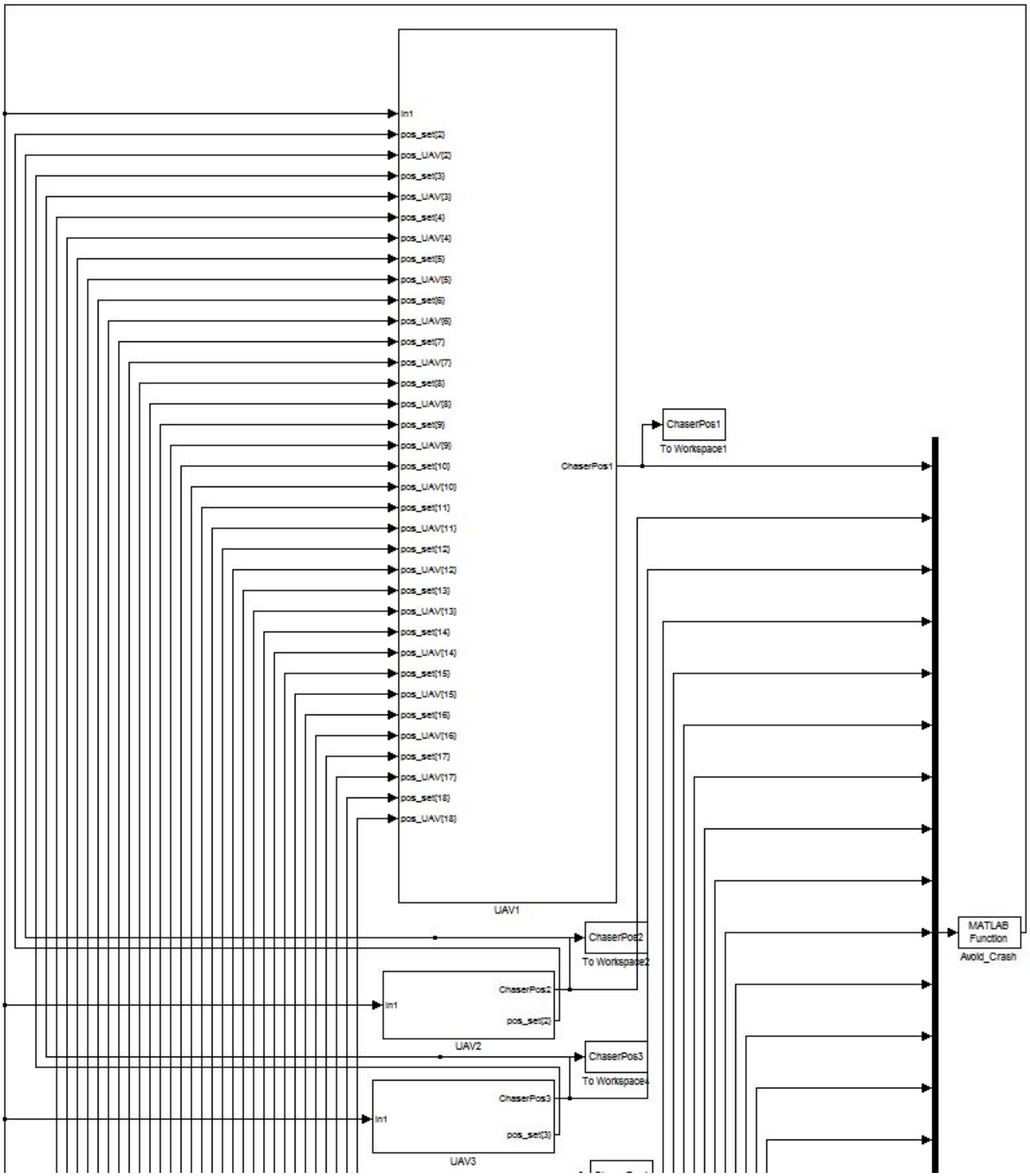


Figure E.3.The 1^{st} level.

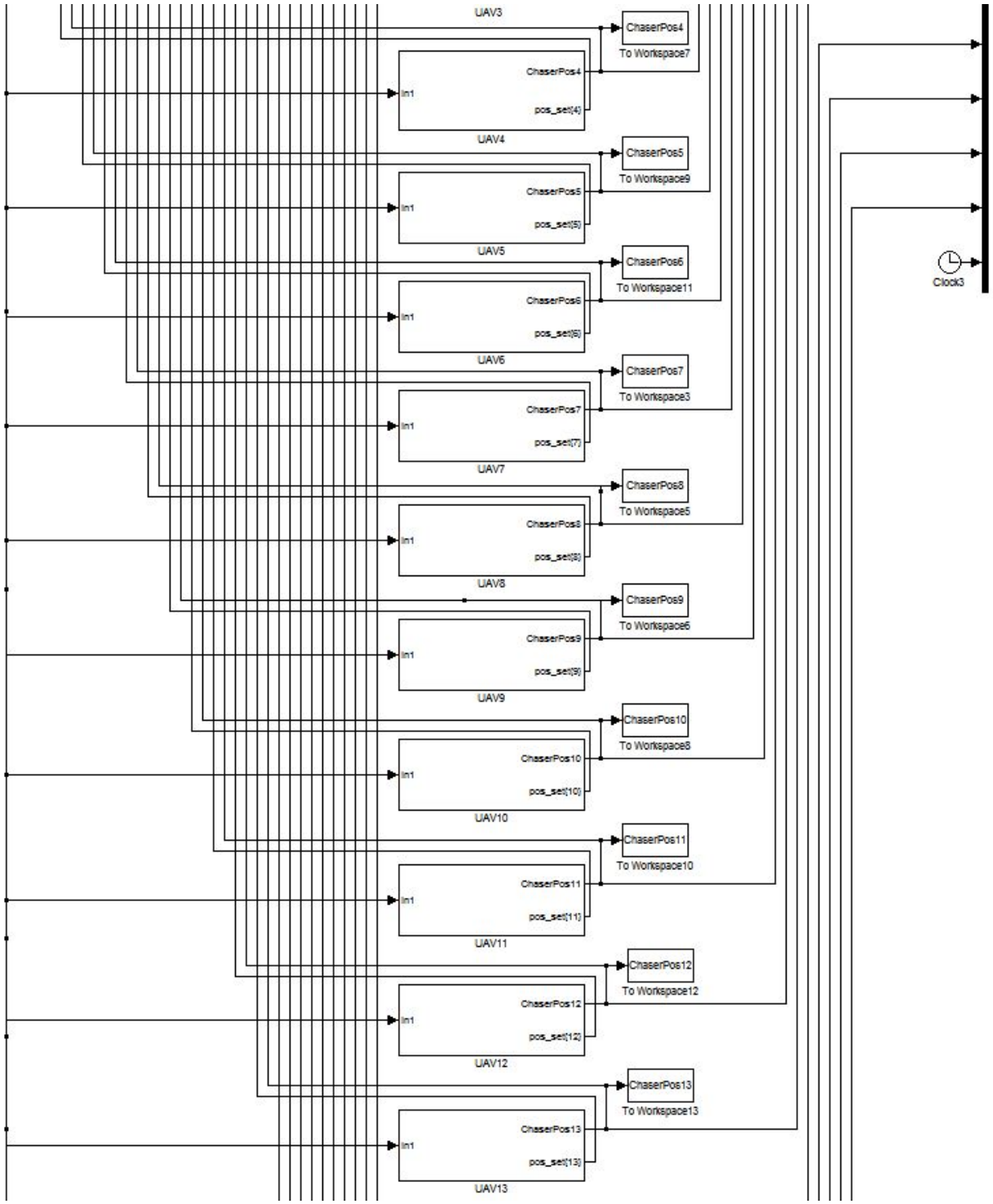


Figure E.4.Continued.

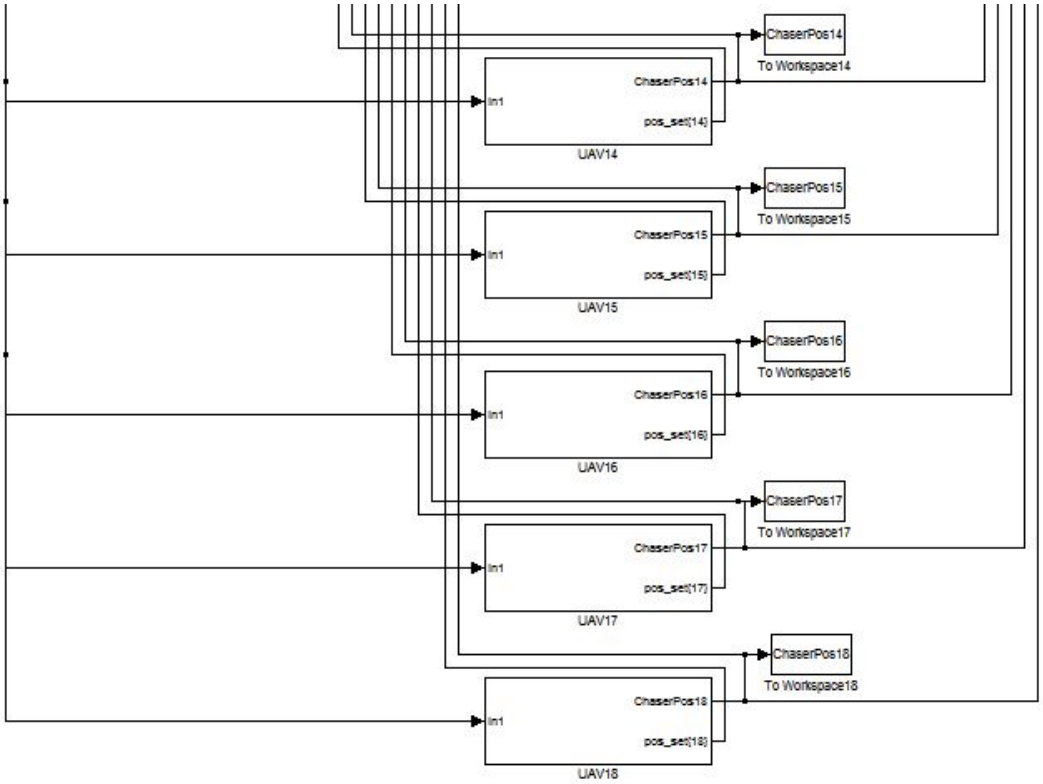


Figure E.5.Continued.

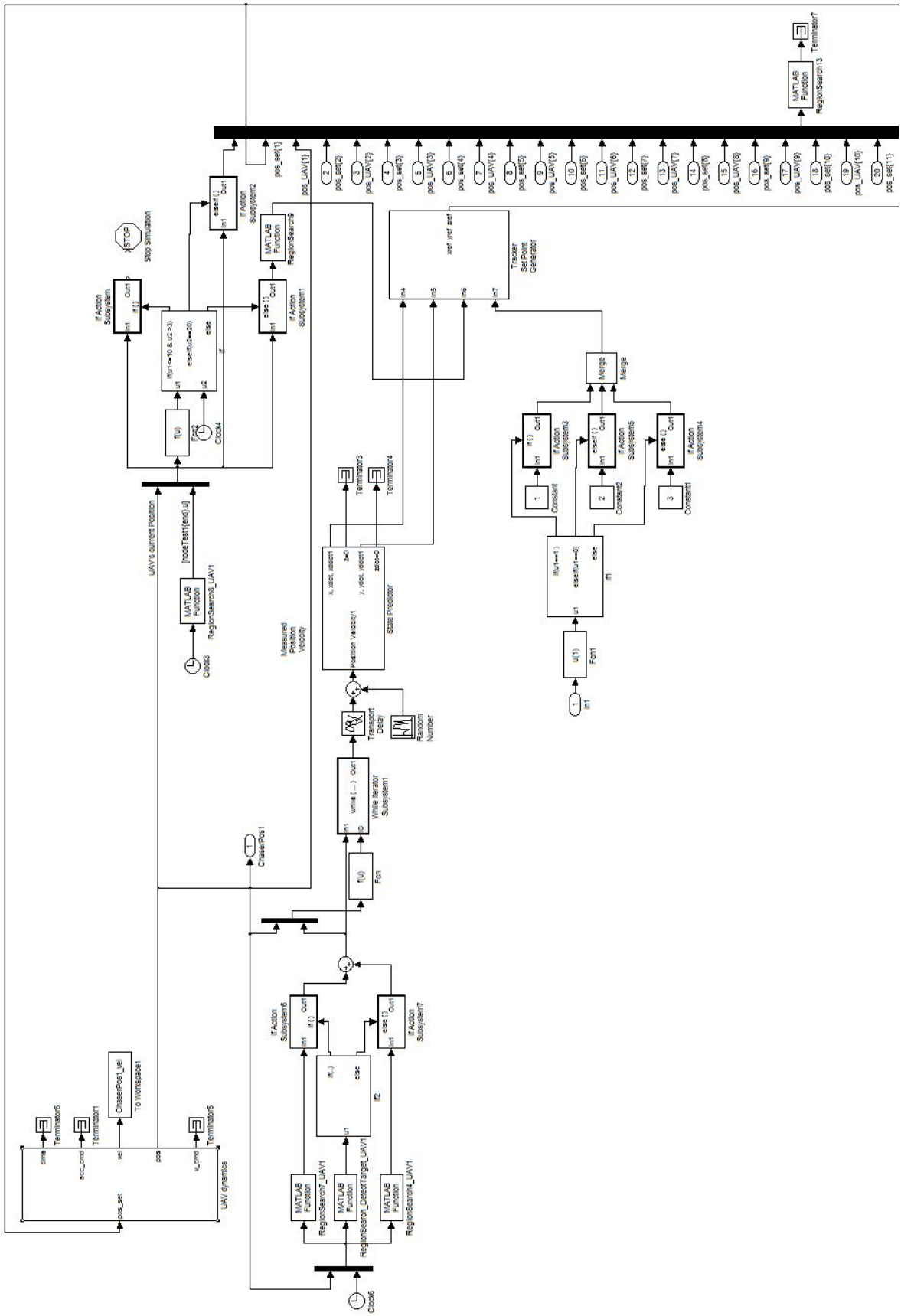


Figure E.6.The 2^{nd} level of UAV_1 (UAV_L) blocks.

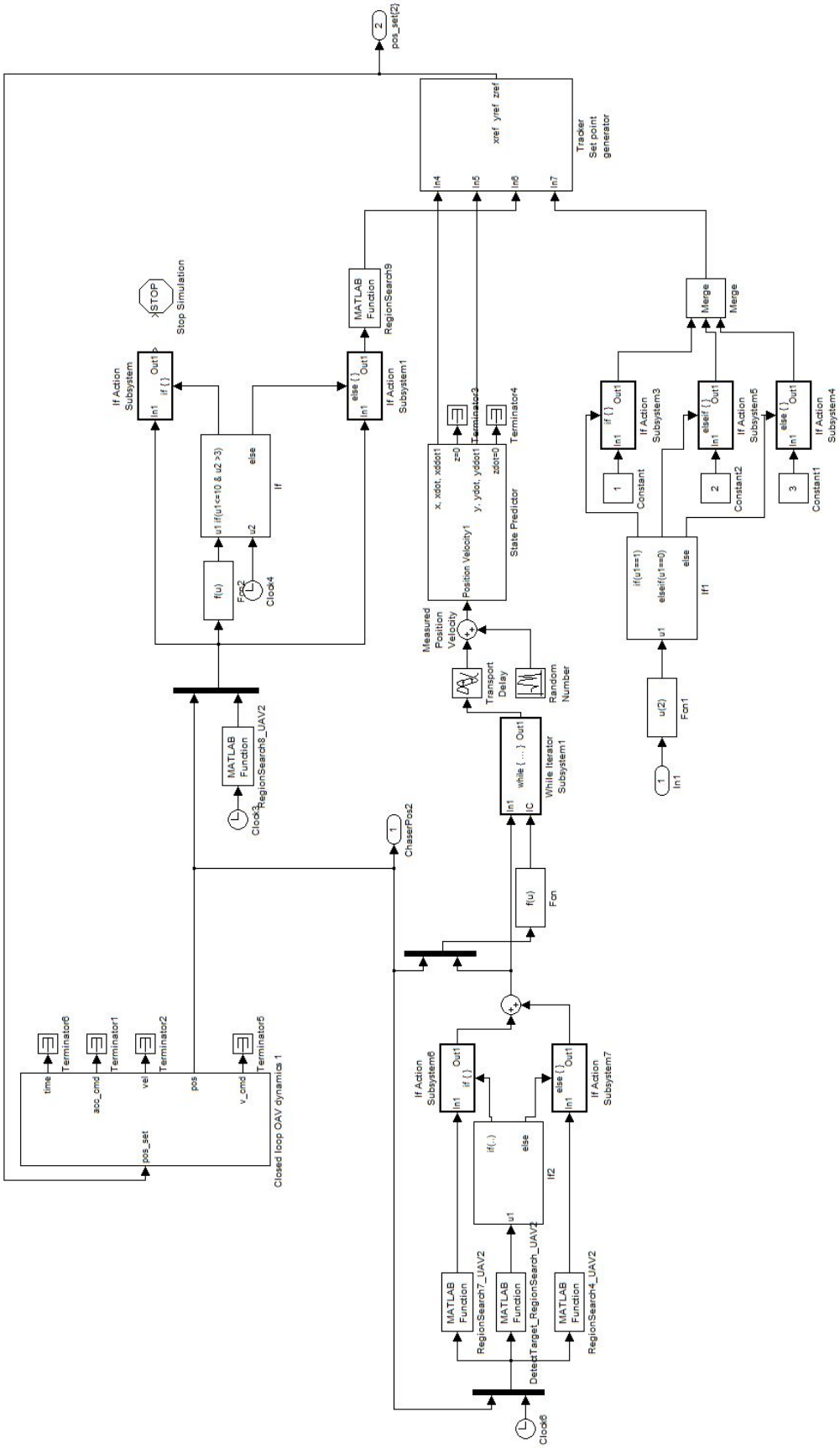


Figure E.7.The 2^{nd} level of UAV_2 (UAV_F) blocks.

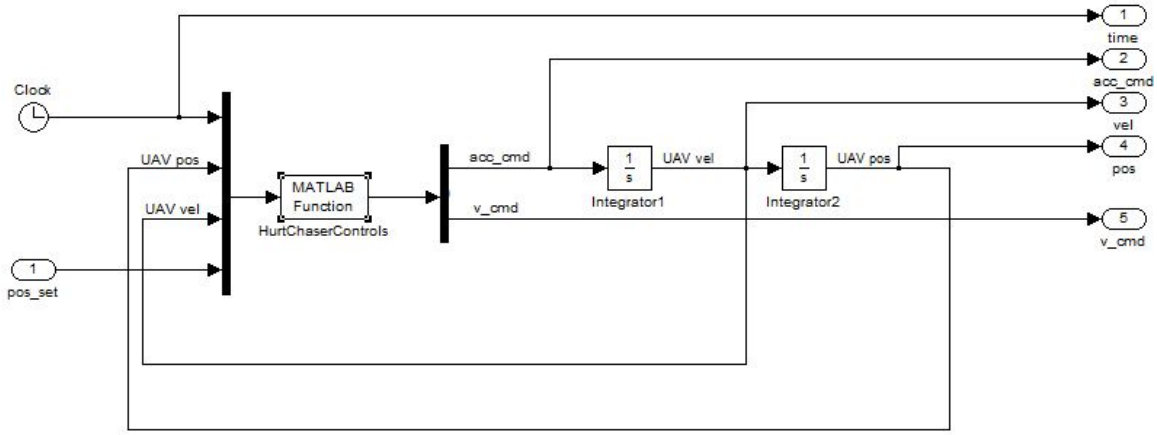


Figure E.8.The 3^{rd} level of UAV dynamics block.

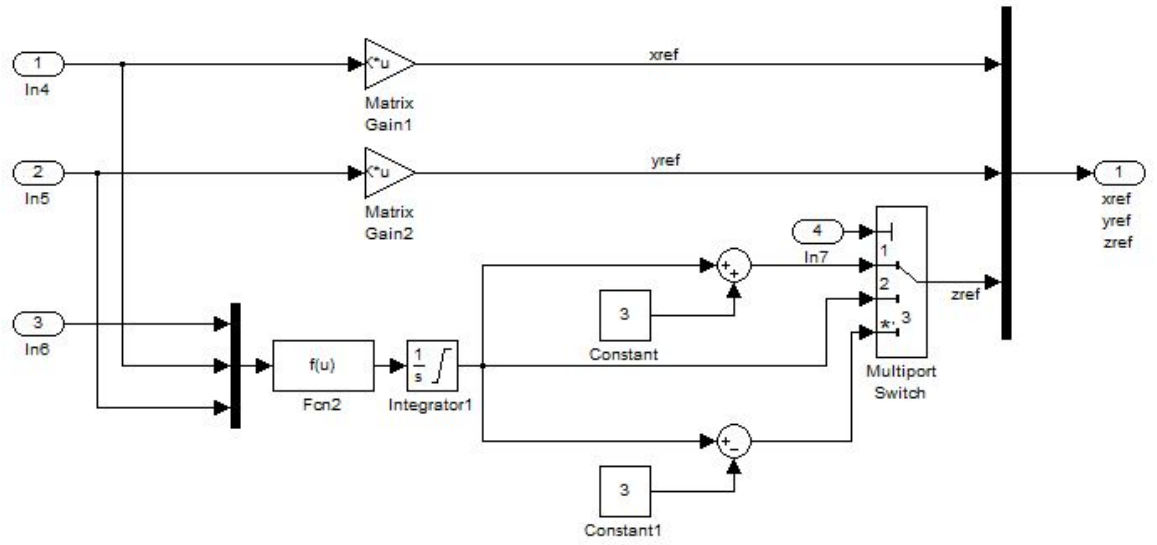


Figure E.9.The 3^{rd} level of Tracker Set Point Generator block.

E.4 MUAVS CAS

CAS built in each UAV maneuvers closely approaching UAVs to change altitudes (CA) or changes velocities (CV) to prevent collisions. If researchers concern about mission completion time, CA (vertical rate of climb of $11.67m/s$, Figure E.10) is preferable than CV since CV delays missions. If small size UAVs are operated in the area with many obstacles, CA might result collisions with obstacles since preplanned trajectories are optimized for 2D path planning. In this case, CV which maintains planned paths is preferable. CV is also preferable since camera images taken at the same altitude assures better target recognition with less image processing computations. It is very unlikely to have more than 3 UAVs are involved in CAS, but if this happens, both CA and CV operate to result the safest flights.

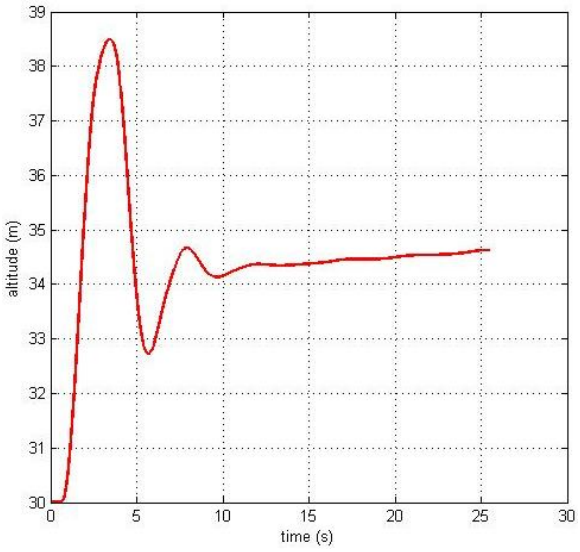


Figure E.10.Sample of altitude change.

E.4.1 Changing Altitude (CA)

Since a gap of $5m$ between UAVs is used, total 6 UAVs can be handled by CAS to avoid collisions without the help of CV according to Figure E.11. 1 UAV stays at Altitude_{normal} and the other 5 UAVs fly either upward or downward.

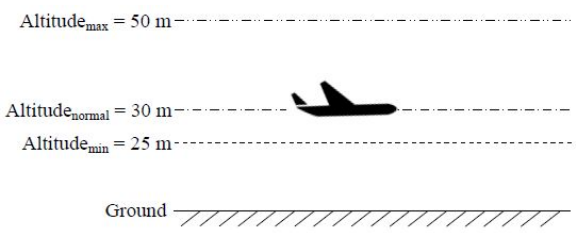


Figure E.11.Altitude$_{max}$, Altitude$_{normal}$, and Altitude$_{min}$ of a fixed-wing UAV according to Table D.1 and D.2.

E.4.2 Changing Velocity (CV)

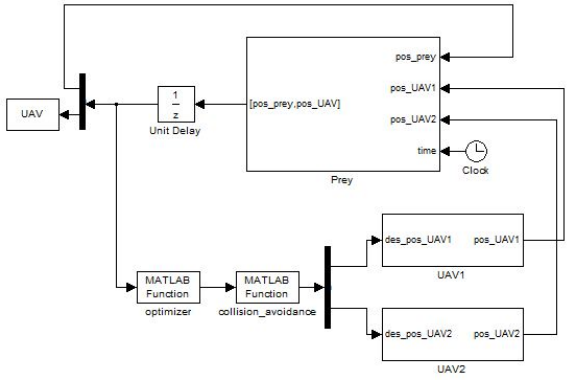
To handle more than 6 UAVs, I need CV which changes velocity as

$$\begin{aligned} v_{1,k} &= v_{1,k} + C\{max(0, \mid P_{1,k}, P_{2,k} \mid - \mid P_{1,k+1}, P_{2,k+1} \mid)\}, \\ v_{2,k} &= v_{2,k} - C\{max(0, \mid P_{1,k}, P_{2,k} \mid - \mid P_{1,k+1}, P_{2,k+1} \mid)\}, \end{aligned} \tag{E.1}$$

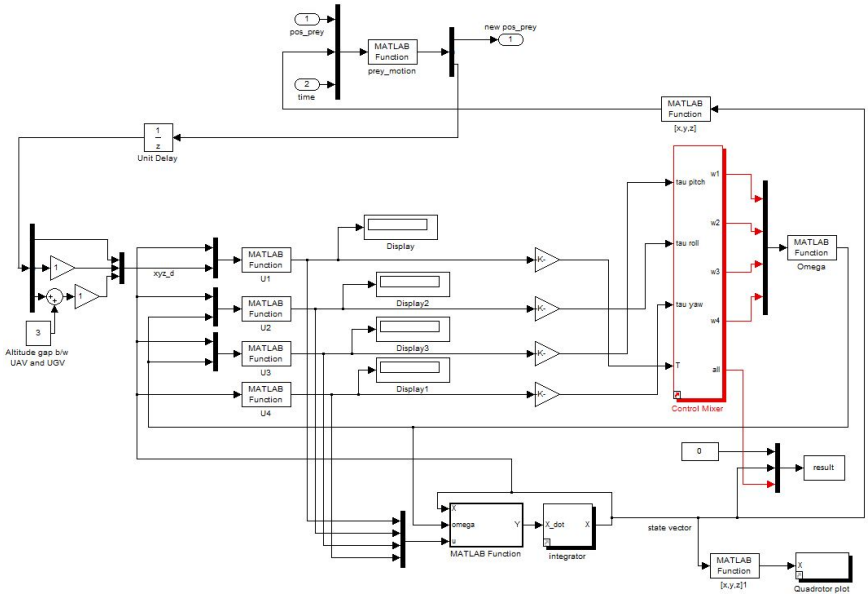
where C is an user-defined constant, $v_{1,k}$ is a velocity of UAV_1 at time step k, $v_{2,k}$ is the velocity of UAV_2 at time step k, $P_{1,k}$ is a position of UAV_1 at time step k, and $P_{2,k}$ is a position of UAV_2 at time step k. All $v_{1,k}$, $v_{2,k}$, $P_{1,k}$, and $P_{2,k}$ are in $\in R^3$.

Appendix F. SIMULINK® Diagrams for the Simulation of UAVs and Preys

Feedback Linearization controller of UAVs is designed using the SIMULINK® [65].



(a) Governing block diagrams of the two preys and UAVs.



(b) Block diagrams of the UAV.

Figure F.1.SIMULINK® diagrmas for the UAVs and preys simulation.

VITA

VITA

Sunghun Jung was born in Seoul, Republic of Korea in 1983. He received the B.S. degree from the University of Minnesota, Twin Cities, in 2009, M.S. degree from the Purdue University, West Lafayette, in 2010, and currently studying Ph.D. degree at the Purdue University. His research interests include control and optimization for autonomous operations of unmanned assets.